AFRL-IF-RS-TR-2000-57
Final Technical Report
April 2000

KNOWLEDGE ENGINEERING WINDOW ON EUROPE – KNOWLEDGE FOUNDATION AND SUPPORT TECHNOLOGIES

University of Edinburgh

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. F103 and K165

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

AFRL-IF-RS-TR-2000-57 has been reviewed and is approved for publication.

APPROVED: William E. Rzepka

WILLIAM E. RZEPKA
Project Engineer

FOR THE DIRECTOR: [illegible]

NORTHRUP FOWLER, Technical Advisor
Information Technology Division
Information Directorate

KNOWLEDGE ENGINEERING WINDOW ON EUROPE – KNOWLEDGE FOUNDATION AND SUPPORT TECHNOLOGIES

John Kingston, Stuart Aitken, and Austin Tate

Contractor: University of Edinburgh
Contract Number: F30602-97-1-0203
Effective Date of Contract: 01 April 1997
Contract Expiration Date: 31 March 2000
Short Title of Work: Knowledge Engineering Window On Europe – Knowledge Foundation and Support Technologies
Period of Work Covered: Apr 97 – Nov 99

Principal Investigator: John Kingston
Phone: (44) 31 650 2732
AFRL Project Engineer: William E. Rzepka
Phone: (315) 330-2762

This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and was monitored by William E. Rzepka, AFRL/IFTD, 525 Brooks Rd, Rome, NY.

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE APRIL 2000	3. REPORT TYPE AND DATES COVERED Final Apr 97 - Nov 99

4. TITLE AND SUBTITLE KNOWLEDGE ENGINEERING WINDOW ON EUROPE - KNOWLEDGE FOUNDATION AND SUPPORT TECHNOLOGIES	5. FUNDING NUMBERS C - F30602-97-1-0203 PE - 62301E PR - IIST TA - 00 WU - 03
6. AUTHOR(S) John Kingston , Stuart Aitken, and Austin Tate	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Edinburgh Artificial Intelligence Applications Institute Division of Informatics 80 South Bridge Edinburgh EH1 1HN United Kingdom	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Defense Advanced Research Project Agency 3701 North Fairfax Drive Arlington VA 22203-1714 Air Force Research Laboratory/IFTD 525 Brooks Road Rome NY 13441-4505	10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2000-57
11. SUPPLEMENTARY NOTES Air Force Research Laboratory Project Engineer: William E. Rzepka/IFTD/(315) 330-2762	
12a. DISTRIBUTION AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

This project brought knowledge and experience of European knowledge engineering methods and techniques to the HPKB program. Originally, this was expected to be done through briefings and training courses, but it quickly became apparent that applying suitable techniques and demonstrating the results was more beneficial to the HPKB program. As a result, AIAI carried out knowledge acquisition for each of the challenge problems and made the results available to the whole HPKB community; implemented methods for representation and reasoning within the Cyc program; and developed a parser and fusion engine in Prolog, which constituted a crucial component of the solution to the Course of Action critiquing challenge problem.

AIAI also pursued the original goals of the Knowledge Engineering Window on Europe project by initial development of a library of problem solving methods; by publishing some Web-based newsletters highlighting techniques and methods of interest to the HPKB community; and by transferring technology to those who attended knowledge acquisition sessions by demonstrating practical application of techniques.

14. SUBJECT TERMS Knowledge Acquisition, Problem Solving Methods, Artificial Intelligence	15. NUMBER OF PAGES 262
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Abstract

This project brought knowledge and experience of European knowledge engineering methods and techniques to the HPKB program. Originally, this was expected to be done through briefings and training courses, but it quickly became apparent that applying suitable techniques and demonstrating the results was more beneficial to the HPKB program. As a result, AIAI carried out knowledge acquisition for each of the challenge problems and made the results available to the whole HPKB community; implemented methods for representation and reasoning within the Cyc program; and developed a parser and fusion engine in Prolog, which constituted a crucial component of the solution to the Course of Action critiquing challenge problem.

AIAI also pursued the original goals of the Knowledge Engineering Window on Europe project by initial development of a library of problem solving methods; by publishing some Web-based newsletters highlighting techniques and methods of interest to the HPKB community; and by transferring technology to those who attended knowledge acquisition sessions by demonstrating practical application of techniques.

Acknowledgements

This research was sponsored by the Defense Advanced Research Projects Agency of the Department of Defense, and was monitored by David Gunning and Murray Burke.

The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government. The United States Government is authorised to reproduce and distribute reprints of this paper and each of the appendices for government purposes notwithstanding any copyright notation hereon.

The KEWE project began in 1997. Since that time the following people have participated directly: Stuart Aitken, Ian Filby, John Kingston and Austin Tate. Associated research has been carried out by Bruce Eddy and Dimitros Sklavadis.

Additional resources for the KEWE project have been provided by the Artificial Intelligence Applications Institute at the University of Edinburgh.

Contents

Appendices: Attached Papers

1 Executive Summary

This project brought knowledge and experience of European knowledge engineering methods and techniques to the HPKB program. Originally, this was expected to be done through briefings and training courses, but it quickly became apparent that applying suitable techniques and demonstrating the results was more beneficial to the HPKB program. As a result, AIAI re-focused its efforts on supporting the development of solutions to the Challenge problems. The following support was provided:

Workaround planning challenge problem: AIAI performed knowledge acquisition to determine the processes involved in selecting workarounds, and published the results to all HPKB participants. AIAI also developed a "proof of concept" system in Cyc that was able to generate some workaround plans, using only Cyc's facilities for ontology representation and declarative theorem proving.

Movement Analysis challenge problem: AIAI performed knowledge acquisition to determine the processes involved in selecting workarounds, and published the results to all HPKB participants.

Crisis management challenge problem: AIAI performed knowledge acquisition in Year 1 to help determine factors that might affect decision making in crises, and in ~~Year 2~~ to determine how interests and actions might interact in a crisis.

Course of action critiquing challenge problem: AIAI developed a parser that converted statements about a COA, written in a structured grammar, into a machine-readable format (specifically, Cyc's MELD language). AIAI also developed a fusion engine that merged MELD statements generated from a text-based COA description with statements generated by Teknowledge's COA sketch tool.

AIAI also pursued the original goals of the Knowledge Engineering Window on Europe project by initial development of a library of problem solving methods; by publishing some Web-based newsletters highlighting techniques and methods of interest to the HPKB community; and by transferring technology to those who attended knowledge acquisition sessions by demonstrating practical application of techniques.

2 Introduction

The design and development of high performance knowledge bases is much more than a programming task. While it is important to have good programming techniques to enable efficient storage and access of knowledge bases running into tens of thousands of axioms, the creation of these knowledge bases requires a range of activities and a range of skills from both human and automated agents. These activities include:

- Knowledge identification: determing where knowledge is being applied, and where that knowledge can be obtained from.
- Knowledge acquisition: collecting the knowledge from any suitable source, and distilling the important knowledge from the supporting information.
- Knowledge representation: describing the knowledge in a manner that is accurate, perspicacious, and yet comprehensible to humans.
- Knowledge based reasoning: linking a inference engine to procedural knowledge to allow incoming data to trigger knowledge-based deductions.

AIAI have supported the work of other participants in the HPKB program by carrying out each of the above activities, as and where necessary. AIAI have targeted those areas of work that can be addressed using knowledge engineering methods or techniques that are more familiar, or better developed, in Europe than in (most of) the USA; hence, we have fulfilled our original goal of making HPKB participants more aware of European knowledge engineering methods and techniques by applying these techniques to solving appropriate components of the Challenge problems.

This report describes the work carried out under the KEWE project. We have:

- carried out knowledge acquisition for three different challenge problems, and documented the results;
- devised a "proof of concept" solution to the Workarounds challenge problem, which uses Cyc for both knowledge representation and reasoning;
- developed a robust and effective parser that takes Course of Action statements in an appropriate grammar and converts them to Cyc axioms;
- developed a fusion engiune that merges these Cyc axioms with axioms derived from a sketch of the same COA (represented in Teknowledge's sketch tool);
- developed and tested an approach for reprersenting problem solving methods in Cyc;
- discussed and developed a format for representing a library of problem solving methods;
- provided some "awareness newsletters" of knowledge engineering activities within Europe.

We also carried out some investigations into using Cyc to support natural language disambiguation; reported on the adequacy of the upper level ontology in Cyc for representing knowledge; and drew up a report highlighting differences between a European method for knowledge acquisition and engineering (CommonKADS) and a US method designed for similar purposes (EXPECT).

2.1 Original Aims

In our original proposal, our aims were expressed as follows:

> AIAI propose to offer support to HPKB program management and participants in leveraging European work on methods and techniques for knowledge acquisition and knowledge modelling. This support would include briefings and training in European knowledge modelling techniques; detailed surveys of available techniques for knowledge acquisition and knowledge modelling; active support in knowledge modelling; and the creation of a standardized library of generic problem solving models which would draw on the best recent research from Europe as well as on the initiative itself.

While our efforts did include some activity on all the work packages proposed above, it became clear early in the project that merely providing awareness of methods would not support the efforts of other HPKB participants well. It was agreed that AIAI would concentrate its main efforts on applying suitable techniques and methods, drawn from our experience of European and other knowledge engineering approaches, to developing documents and software that supported solutions to the challenge problems. The work packages and associated deliverables were modified as a result.

The techniques used are described in the following sections.

3 Knowledge Acquisition techniques

A range of **knowledge acquisition** techniques have been used, including interviewing, laddering, card sorting, and "20 Questions".

The techniques that were used include:

- Working through a case study, in which the experts were required to ask verbally for all the information needed to solve the problem, for the Year 1 Movement Analysis challenge problem. The "verbal case study" approach is the essence of the "20 Questions" knowledge elicitation technique.
- Rapid prototyping of knowledge models: for both the Movement Analysis and Workarounds challenge problems, IDEF3 models of the process were created based on initial interviews, and were then dynamically refined by showing them to the experts and inviting constructive criticism. This approach is considered to be an improvement on the "rapid prototyping" approach to development of knowledge based systems, because the knowledge acquisition benefits of swift expert feedback are realised, while the risks of piecemeal system development are avoided.
- "Card sorting" for the crisis management challenge problem, to acquire key properties that differentiate economic, military and political actions. The "20 questions" and "rapid prototyping of models" approaches were also used for the Crisis Management challenge problem in year 1
- Categorisation of interests and actions for the year 2 crisis management challenge problem. This was an experiment in performing knowledge acquisition by deriving information from a book (in this case, Helmut Kahn's "On Thermonuclear War"), re-categorising it, and obtaining constructive criticism by e-mail.

3.1 Deliverables

K-1 : Knowledge acquisition results from Year 1

K-2 : Knowledge acquisition results from Year 2.

These deliverables can be found in Appendices C & D.

4 Knowledge Representation and Reasoning

AIAI have done various pieces of work related to *knowledge representation*, mostly in relation to Cyc. The approaches that have been used include:

- Characterising the capabilities of problem solving methods using a synthesis of existing knowledge representation approaches;
- Comparing and constrating two major knowledge representation systems (EXPECT and Cyc);
- Developing a parser that transforms Course of Action Statements from representation in a pseudo-English grammar to representation in MELD, the language of Cyc, and then merges these statements with the output of Teknowledge's sketch tool;
- Reporting on the different approaches to knowledge representation and reasoning used for the Workarounds challenge problem;
- Attempting to use Cyc's ontology as a basis for a natural language disambiguation system;
- Implementing a conceptual problem solving method for diagnostic problems within Cyc;
- Building a "proof of concept" planning system that was able to tackle parts of the Workarounds challenge problem, using a rich planning ontology and Cyc's theorem proving inference mechanism.

5 Summary of deliverables

Deliverables described in normal font are those originally promised for the KEWE project. Deliverables described in italics were originally promised, but have been revised to accord with the revised aims of the project. Deliverables in bold font are additional deliverables that were not originally promised but which were produced by the revised activities of the project.

Deliverable	Due Date	WP #	Description
B-1	end-Q1/Y1	WP 1	Materials from briefings given
C-1	end-Q4/Y1	WP 2	*Brief summary of work done with CYC*
C-2	end-Q3/Y3	WP 2	*Discussion of Cyc's upper level ontology*
K-1	end-Q4/Y1	WP 3	*Knowledge Models and knowledge acquisition products from year 1*
K-2	end-Q1/Y3	WP 3	*Knowledge Models and knowledge acquisition products (Year II)*
P-1	end-Q2/Y2	WP 4	Code for Y1 Workarounds Challenge Problem
P-2	end-Q2/Y3	WP 4	Code for Y2 COA Challenge problem
P-3	end-Q2/Y2	WP 4	**Report on Y1 Workarounds Challenge Problem**
P-4	end-Q2/Y3	WP 4	**Report on Y2 COA Challenge problem**
P-5	end-Q3-Y3	WP 4	**Paper comparing the 4 approaches to workarounds problem solving**
N-1	end-Q2/Y1	WP 5	Newsletter
N-2	end-Q4/Y1	WP 5	Newsletter
N-3	end-Q4/Y1	WP 5	Newsletter
L-0	end-Q4/Y3	WP 6	**Report comparing two approaches to knowledge engineering: EXPECT and KADS**
L-1	end-Q4/Y1	WP 6	*Report on capability descriptions for problem solving methods*
L-2	end-Q4/Y2	WP 6	*Paper on the implementation of a PSM in Cyc*
M-1	end-Q4/Y3		**Paper/report on techniques attempted for natural language disambiguation in Cyc**
Q-1 to Q-9	each Q	WP 7	Quarterly Progress Reports
Y-1	end-Q4/Y1	WP 7	Year 1 Annual Report
Y-2	end-Q4/Y2	WP 7	Year 2 Annual Report
Y-3	end-Q3/Y3	WP 7	Final Report

The deliverables that are attached to this report consist of all the above except the project management reports: deliverables Q-1 to Q-9, Y-1 and Y-2. To include these deliverables would merely be to repeat information that is contained in this report.

The deliverables can be found in the appendices to this report.

References

[1] Benjamins, R., de Barros, L. N., and Valente, A. Constructing planners through problem solving methods. *Proceedings of the Knowledge Acquisition Workshop 1996*, Banff 1996.

[2] de Barros, L. N., Valente, A., and Benjamins, R. Modelling planning tasks. *Proceedings of AIPS 1996*, pp. 11-18, 1996.

[3] de Barros, L. N., Hendler, J., and Benjamins, R. Par-KAP: A knowledge acquisition tool for building practical planning systems. *Proceedings of IJCAI 1997*, pp. 1246–1251, 1997.

[4] H. Kahn, "On Thermonuclear War", 2nd edition, Greenwood Publishing Group, 1978.

[5] J. Kingston, N. Shadbolt and A. Tate, "CommonKADS Models for Knowledge Based Planning", Proceedings of AAAI-96, Portland, Oregon, August 1996.

[6] O-Plan Project Team *Task Formalism Manual v 2.3* AIAI, July 1995.

[7] U.S. Army Engineer School *Engineer Systems Handbook* April 1997.

[8] U.S. Army *Engineer Field Data FM 5-34* July 1997.

[9] Valente, A. Knowledge-level analysis of planning systems. *SIGART Bulletin*, Vol. 6, No. 1, pp. 33–41, 1995.

Appendices

Awareness deliverables

APPENDIX A: Briefings

APPENDIX B: Newsletters Deliverables N-1, N-2: Newsletters prepared and published,

Knowledge Acquisition Deliverables

APPENDIX C: Knowledge Acquisition Results (Year 1) Deliverable K-1: Knowledge Models and results of knowledge acquisition (Year 1).

APPENDIX D: Knowledge Acquisition Results (Year 2) Deliverable K-2: Results of knowledge acquisition (Year 2).

Knowledge Representation and Reasoning Deliverables

APPENDICES E-G: Knowledge Representation and Reasoning: Problem Solving Methods L-0 (EXPECT vs KADS), L-1 (Capabilities of PSMs), L-2 (a PSM in Cyc)

APPENDICES H-L: Knowledge representation and Reasoning: Challenge problems Deliverable P-5 (paper about 4 approaches to workarounds), P-1 and P-3 (Workarounds problem solver), P-2 and P-4 (Parser and fusion engine)

APPENDIX M-O: Knowledge Representation and Reasoning in Cyc Deliverables C-1 (Brief summary of work done with Cyc), C-2 (Report on Cyc's upper ontology), M-1 (Summary of a Natural Language Disambiguation research project using Cyc)

A Briefings given

The briefings included in Deliverable B-1 were presented to various HPKB PI (Principal Investigator) meetings. These meetings were as follows:

- HPKB Kick-off Meeting, Staunton, Virginia, June 1997;
- PI meeting, San Diego, California, December 1997;
- Year 1 meeting, Washington DC, July 1998;
- Year 1.5 meeting, Austin, Texas, January 1999;
- Final meeting, Washington DC, October 1999

The briefings are presented in the following pages, in chronological order.

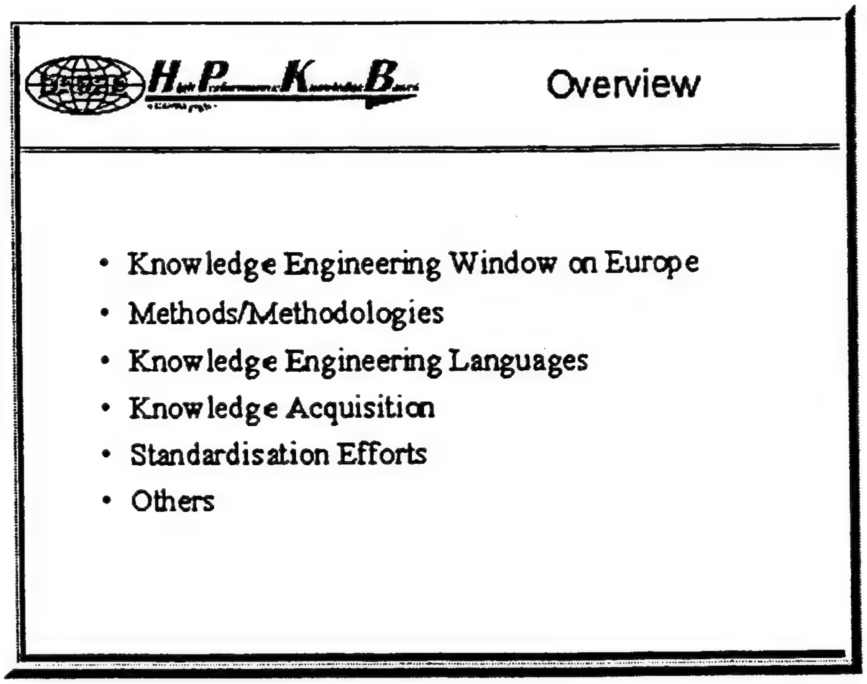
A.1 Knowledge Engineering Window on Europe: Initial Briefing

Publication details: Briefing presented at HPKB Kick-off Meeting, Staunton, Virginia, June 1997

John Kingston

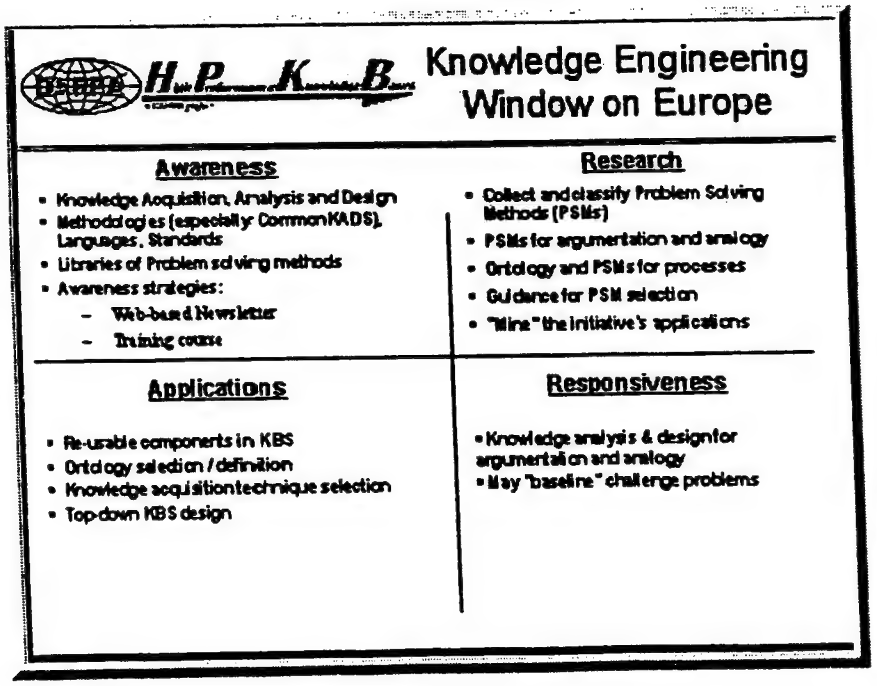
Purpose: To describe AIAI's intended work packages for HPKB, providing a "Knowledge Engineering Window on Europe" through briefings, awareness seminars, and newsletters.

Abstract: AIAI proposed to act as a *Window on Europe* for the HPKB program, to avoid duplication of previous research and to utilize the best practices which were already publicly available through European research.



Overview

- Knowledge Engineering Window on Europe
- Methods/Methodologies
- Knowledge Engineering Languages
- Knowledge Acquisition
- Standardisation Efforts
- Others



Knowledge Engineering Window on Europe

Awareness

- Knowledge Acquisition, Analysis and Design
- Methodologies (especially CommonKADS), Languages, Standards
- Libraries of Problem solving methods
- Awareness strategies:
  - Web-based Newsletter
  - Training course

Research

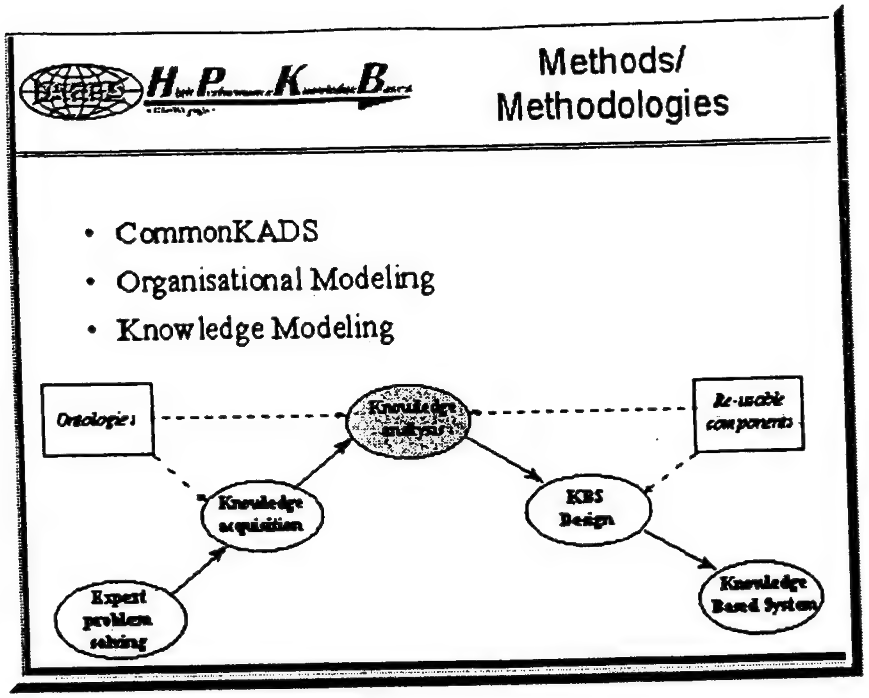
- Collect and classify Problem Solving Methods (PSMs)
- PSMs for argumentation and analogy
- Ontology and PSMs for processes
- Guidance for PSM selection
- "Mine" the initiative's applications

Applications

- Re-usable components in KBS
- Ontology selection / definition
- Knowledge acquisition technique selection
- Top-down KBS design

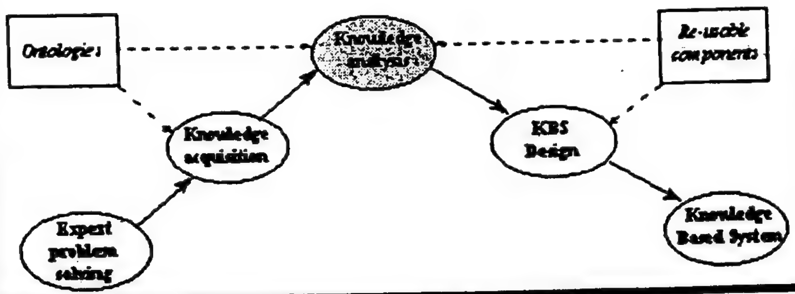
Responsiveness

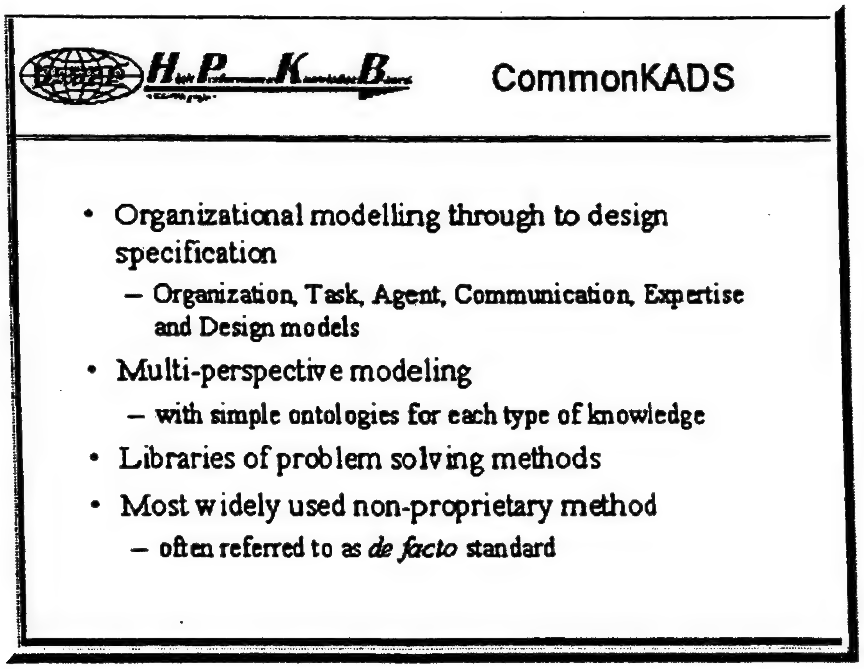
- Knowledge analysis & design for argumentation and analogy
- May "baseline" challenge problems



Methods/ Methodologies

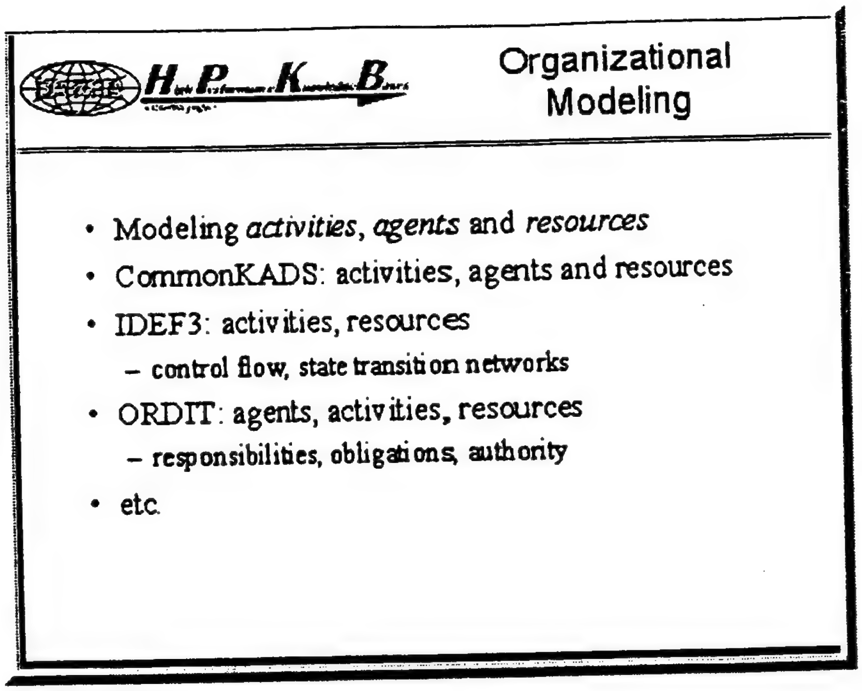
- CommonKADS
- Organisational Modeling
- Knowledge Modeling





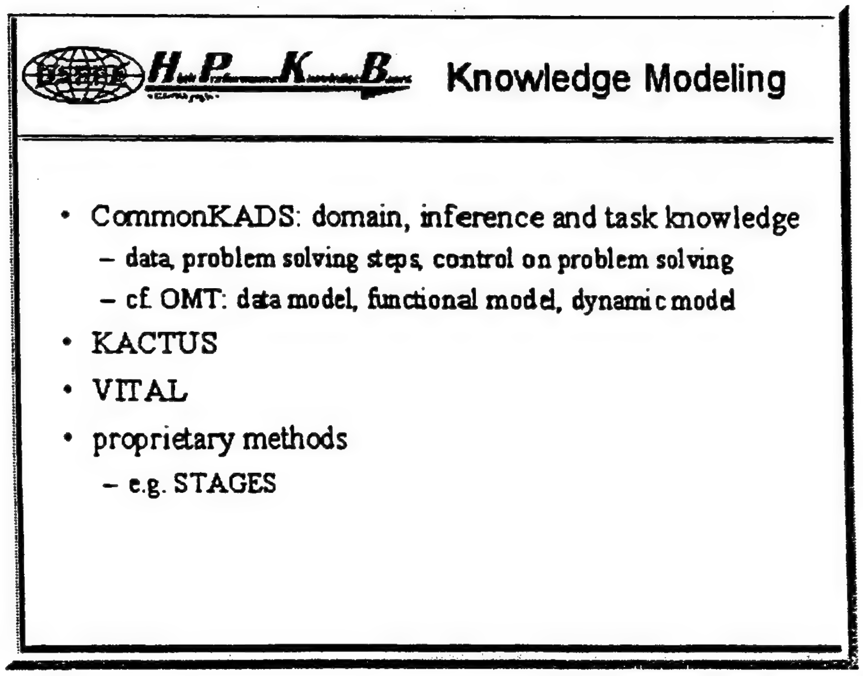
CommonKADS

- Organizational modelling through to design specification
  - Organization, Task, Agent, Communication, Expertise and Design models
- Multi-perspective modeling
  - with simple ontologies for each type of knowledge
- Libraries of problem solving methods
- Most widely used non-proprietary method
  - often referred to as *de facto* standard



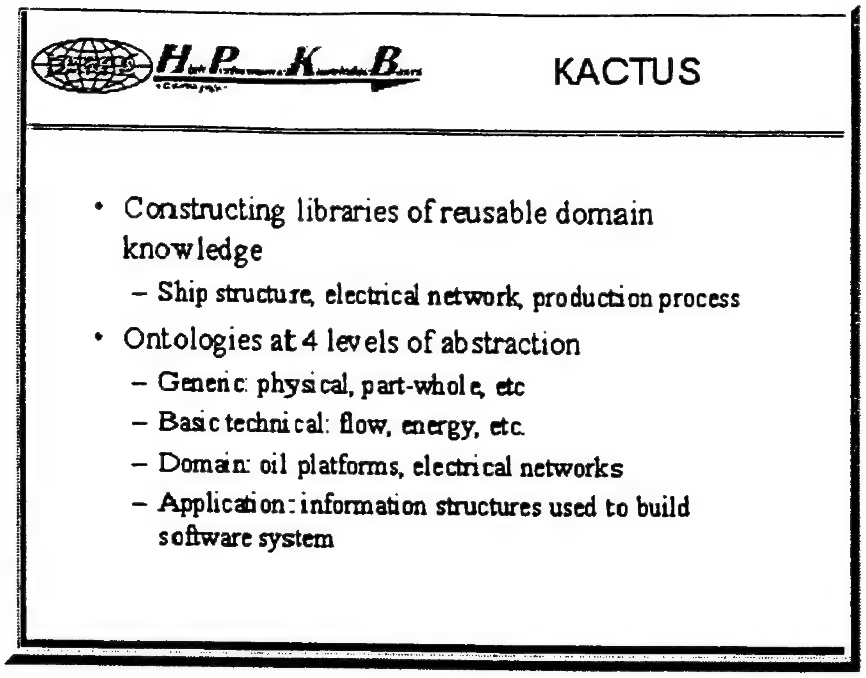
Organizational Modeling

- Modeling *activities*, *agents* and *resources*
- CommonKADS: activities, agents and resources
- IDEF3: activities, resources
  - control flow, state transition networks
- ORDIT: agents, activities, resources
  - responsibilities, obligations, authority
- etc.



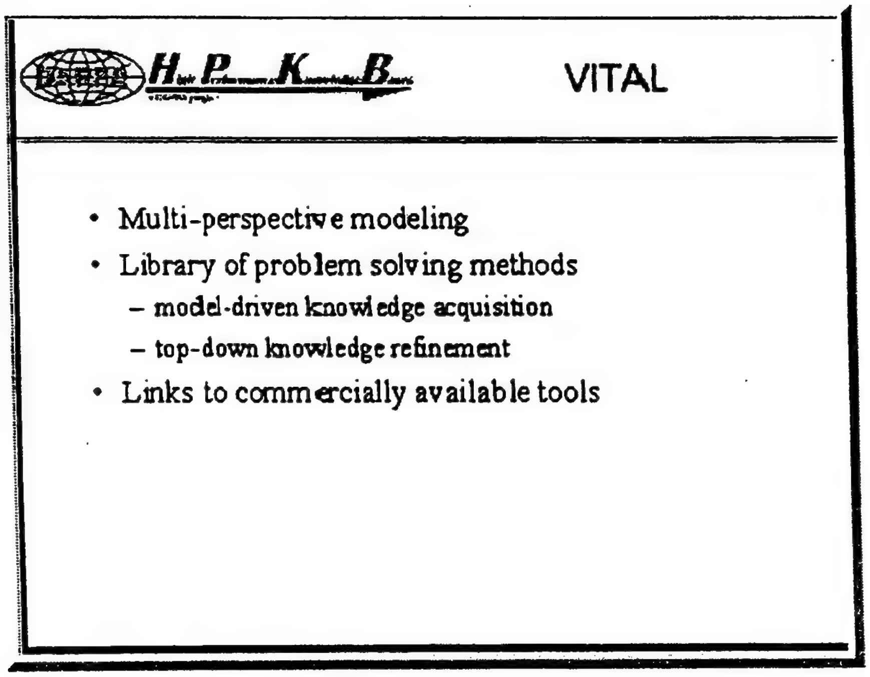
Knowledge Modeling

- CommonKADS: domain, inference and task knowledge
  - data, problem solving steps, control on problem solving
  - cf. OMT: data model, functional model, dynamic model
- KACTUS
- VITAL
- proprietary methods
  - e.g. STAGES



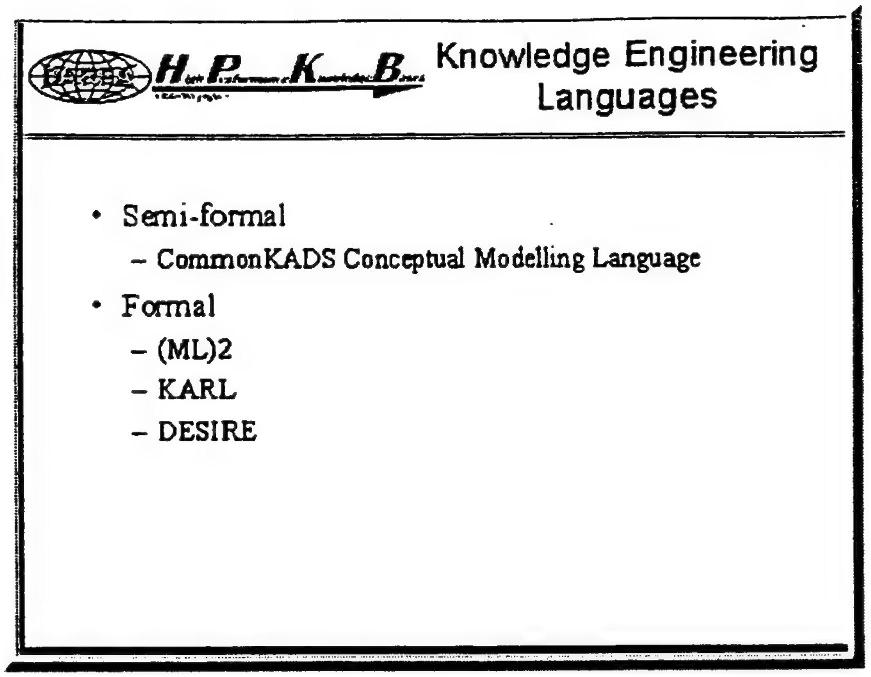
KACTUS

- Constructing libraries of reusable domain knowledge
  - Ship structure, electrical network, production process
- Ontologies at 4 levels of abstraction
  - Generic: physical, part-whole, etc
  - Basic technical: flow, energy, etc.
  - Domain: oil platforms, electrical networks
  - Application: information structures used to build software system



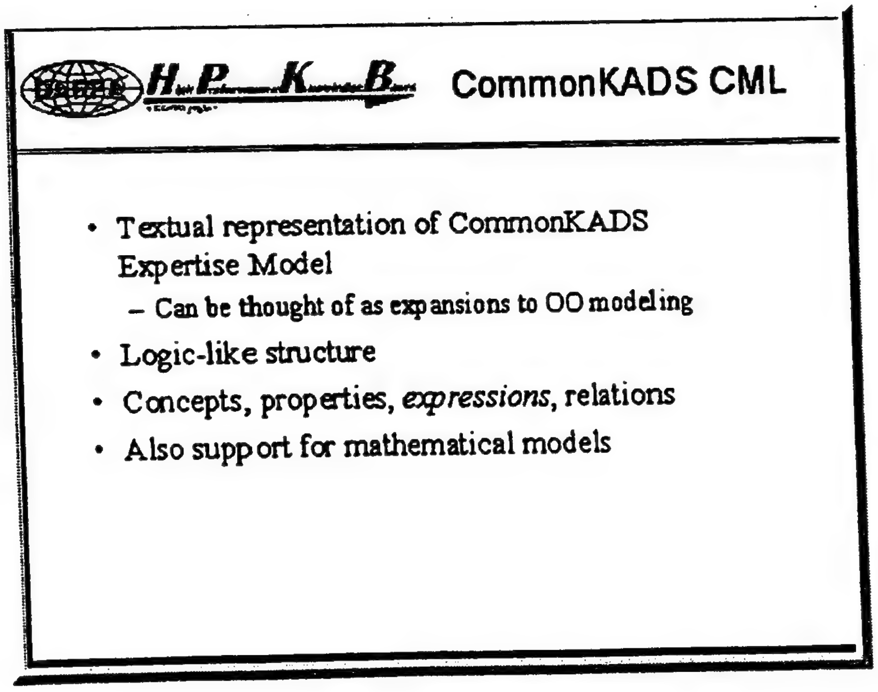
VITAL

- Multi-perspective modeling
- Library of problem solving methods
  - model-driven knowledge acquisition
  - top-down knowledge refinement
- Links to commercially available tools



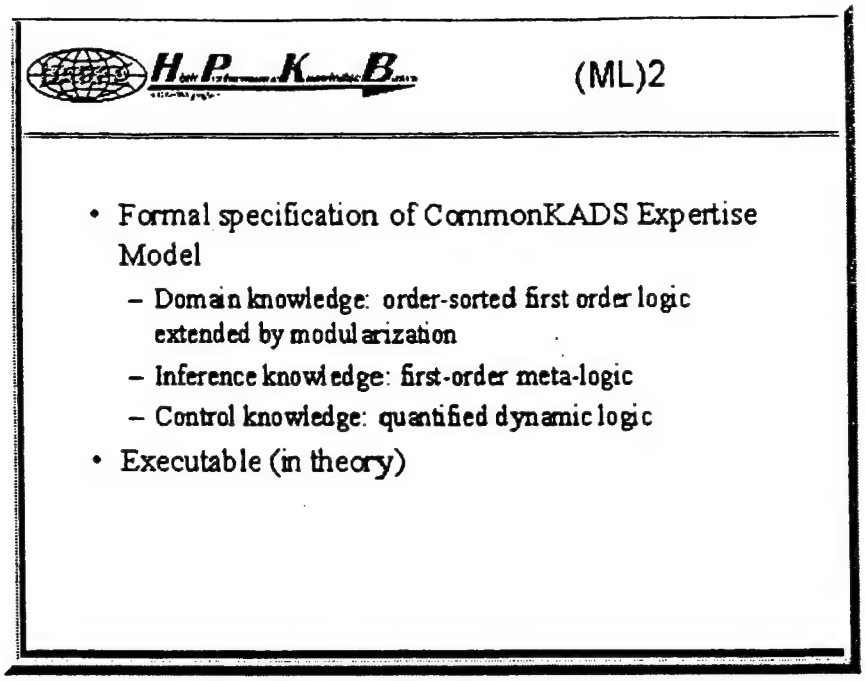
Knowledge Engineering Languages

- Semi-formal
  - CommonKADS Conceptual Modelling Language
- Formal
  - (ML)2
  - KARL
  - DESIRE



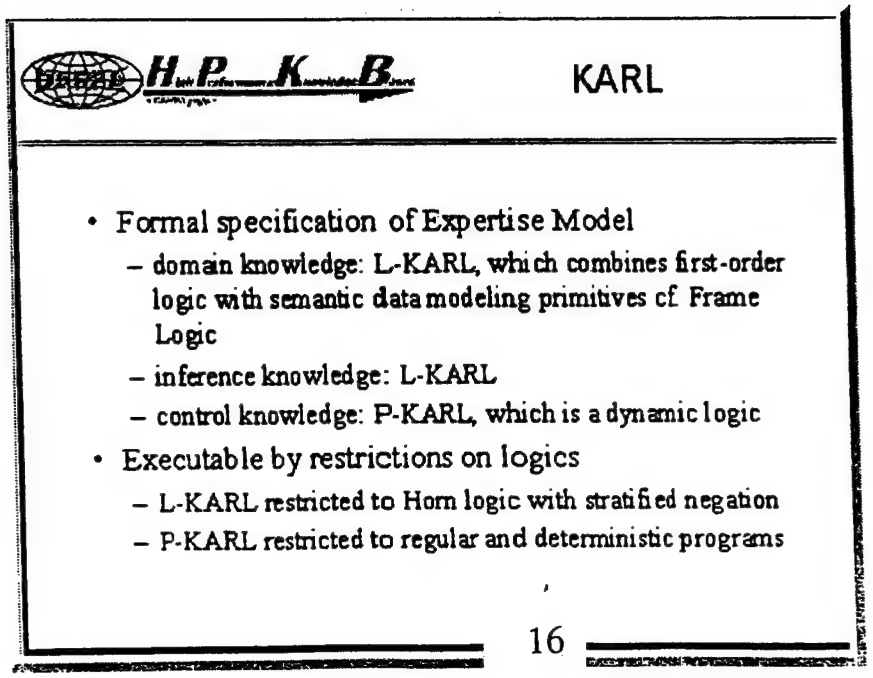
CommonKADS CML

- Textual representation of CommonKADS Expertise Model
  - Can be thought of as expansions to OO modeling
- Logic-like structure
- Concepts, properties, *expressions*, relations
- Also support for mathematical models



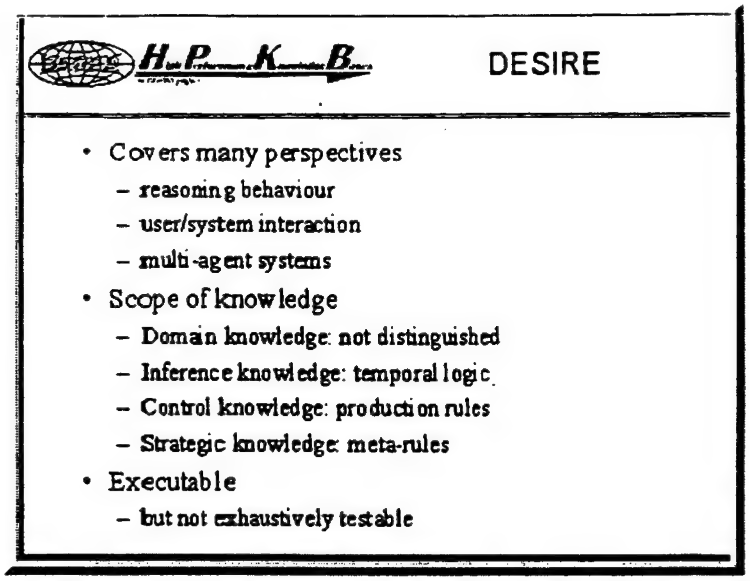
(ML)2

- Formal specification of CommonKADS Expertise Model
  - Domain knowledge: order-sorted first order logic extended by modularization
  - Inference knowledge: first-order meta-logic
  - Control knowledge: quantified dynamic logic
- Executable (in theory)



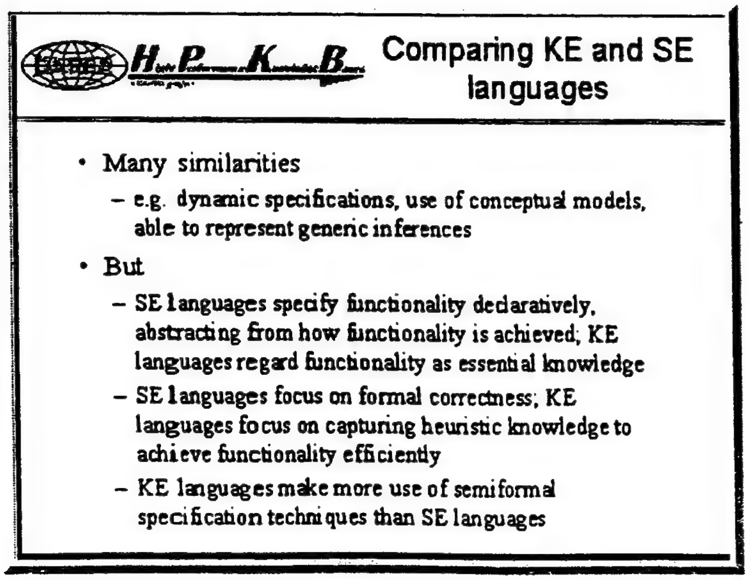
KARL

- Formal specification of Expertise Model
  - domain knowledge: L-KARL, which combines first-order logic with semantic data modeling primitives cf. Frame Logic
  - inference knowledge: L-KARL
  - control knowledge: P-KARL, which is a dynamic logic
- Executable by restrictions on logics
  - L-KARL restricted to Horn logic with stratified negation
  - P-KARL restricted to regular and deterministic programs



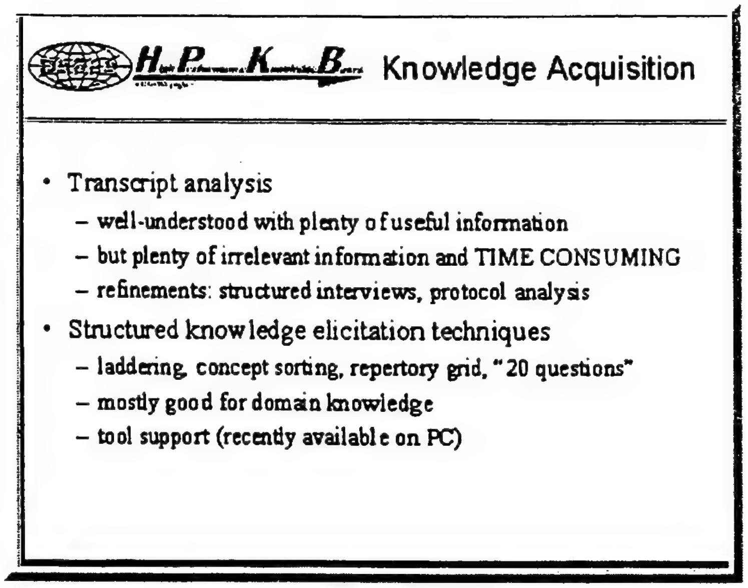
DESIRE

- Covers many perspectives
  - reasoning behaviour
  - user/system interaction
  - multi-agent systems
- Scope of knowledge
  - Domain knowledge: not distinguished
  - Inference knowledge: temporal logic
  - Control knowledge: production rules
  - Strategic knowledge: meta-rules
- Executable
  - but not exhaustively testable



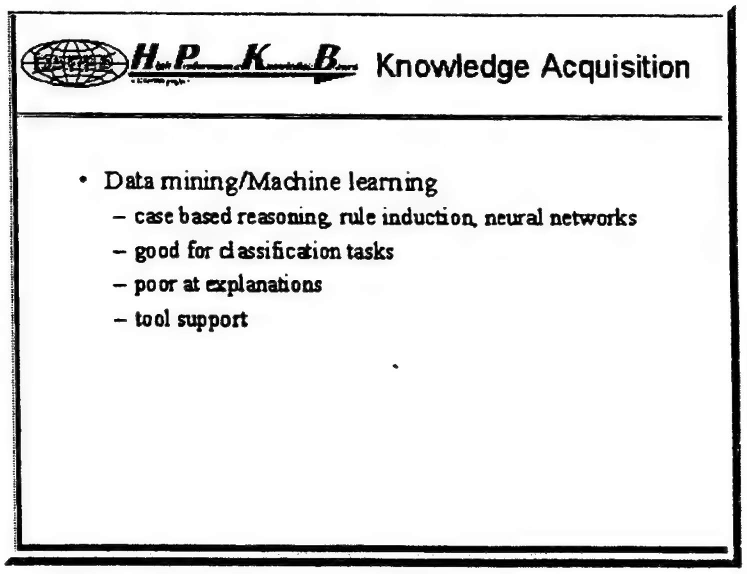
Comparing KE and SE languages

- Many similarities
  - e.g. dynamic specifications, use of conceptual models, able to represent generic inferences
- But
  - SE languages specify functionality declaratively, abstracting from how functionality is achieved; KE languages regard functionality as essential knowledge
  - SE languages focus on formal correctness; KE languages focus on capturing heuristic knowledge to achieve functionality efficiently
  - KE languages make more use of semiformal specification techniques than SE languages



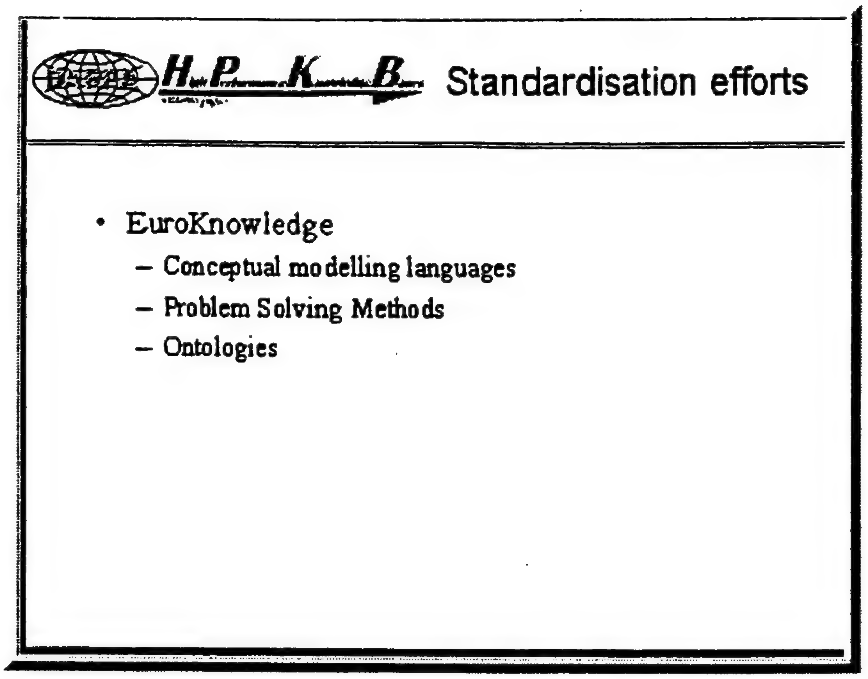
H.P.K.B. Knowledge Acquisition

- Transcript analysis
  - well-understood with plenty of useful information
  - but plenty of irrelevant information and TIME CONSUMING
  - refinements: structured interviews, protocol analysis
- Structured knowledge elicitation techniques
  - laddering, concept sorting, repertory grid, "20 questions"
  - mostly good for domain knowledge
  - tool support (recently available on PC)



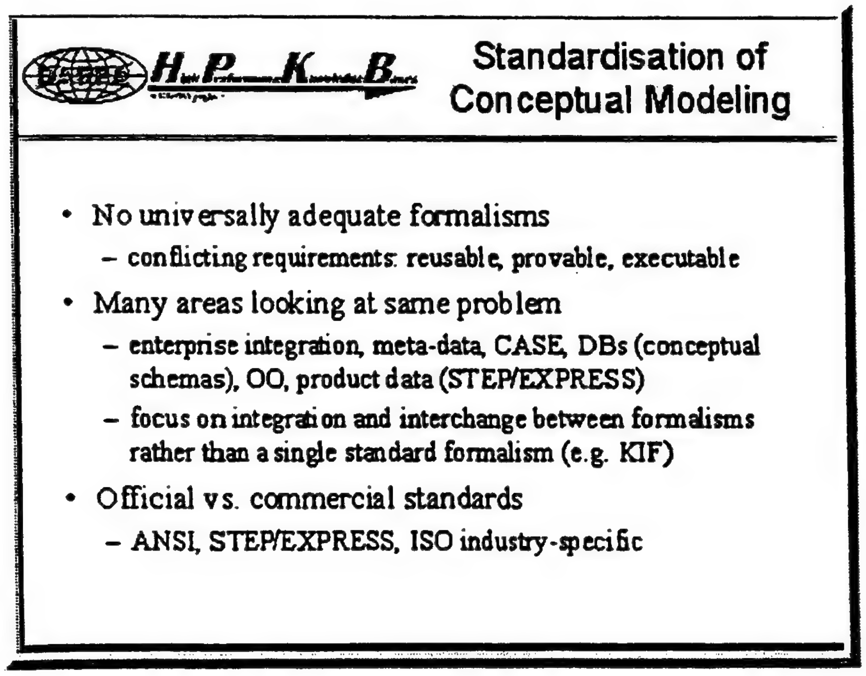
H.P.K.B. Knowledge Acquisition

- Data mining/Machine learning
  - case based reasoning, rule induction, neural networks
  - good for classification tasks
  - poor at explanations
  - tool support



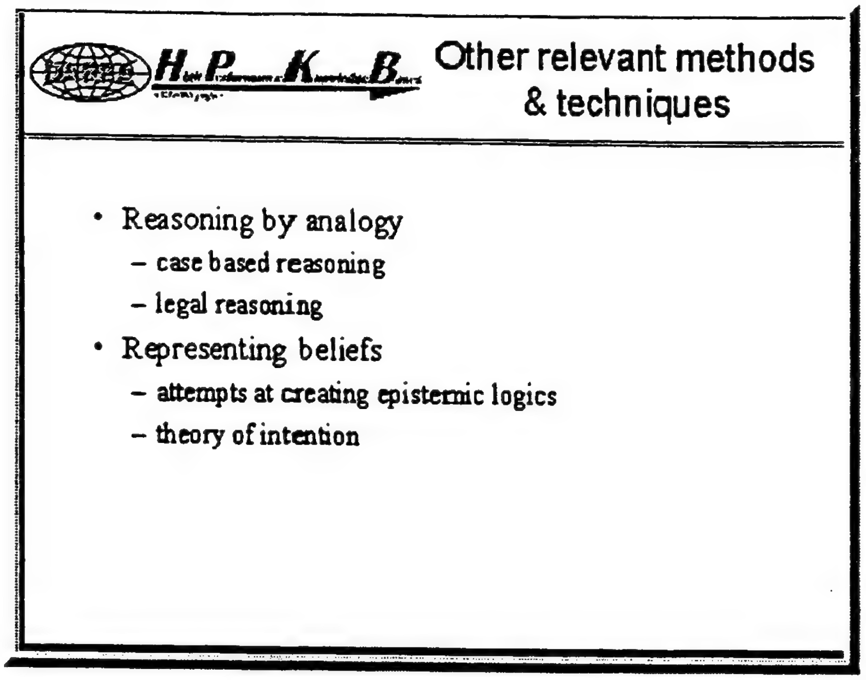
Standardisation efforts

- EuroKnowledge
  - Conceptual modelling languages
  - Problem Solving Methods
  - Ontologies



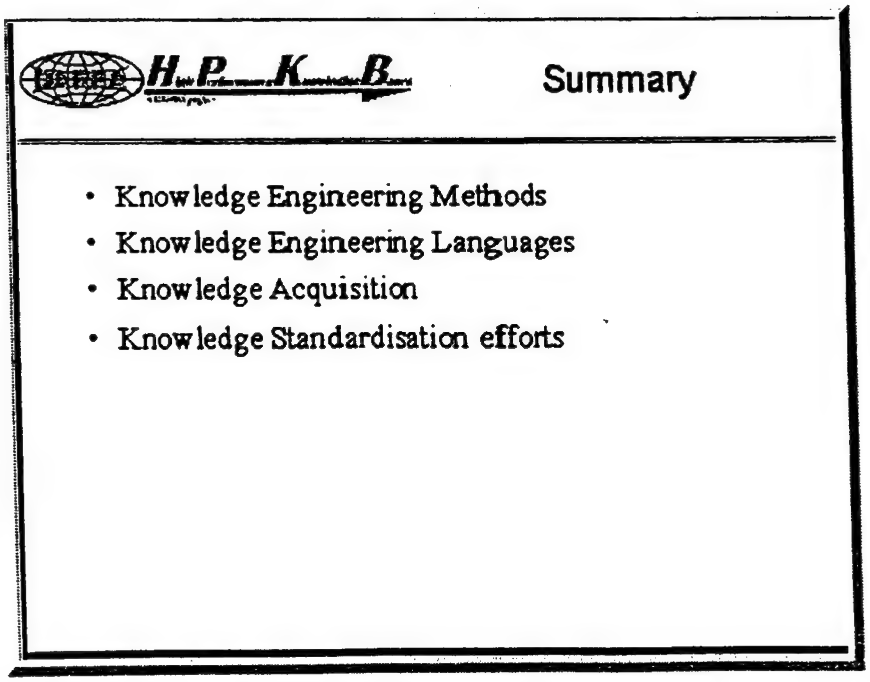
Standardisation of Conceptual Modeling

- No universally adequate formalisms
  - conflicting requirements: reusable, provable, executable
- Many areas looking at same problem
  - enterprise integration, meta-data, CASE, DBs (conceptual schemas), OO, product data (STEP/EXPRESS)
  - focus on integration and interchange between formalisms rather than a single standard formalism (e.g. KIF)
- Official vs. commercial standards
  - ANSI, STEP/EXPRESS, ISO industry-specific



Other relevant methods & techniques

- Reasoning by analogy
  - case based reasoning
  - legal reasoning
- Representing beliefs
  - attempts at creating epistemic logics
  - theory of intention



Summary

- Knowledge Engineering Methods
- Knowledge Engineering Languages
- Knowledge Acquisition
- Knowledge Standardisation efforts

A.2 Revised plan and actions

Publication details: Briefing presented at HPKB Kick-off Meeting, Staunton, Virginia, December 1997

John Kingston

Purpose: To describe AIAI's activities to date as well as revisions to AIAI's aims and objectives on the HPKB program

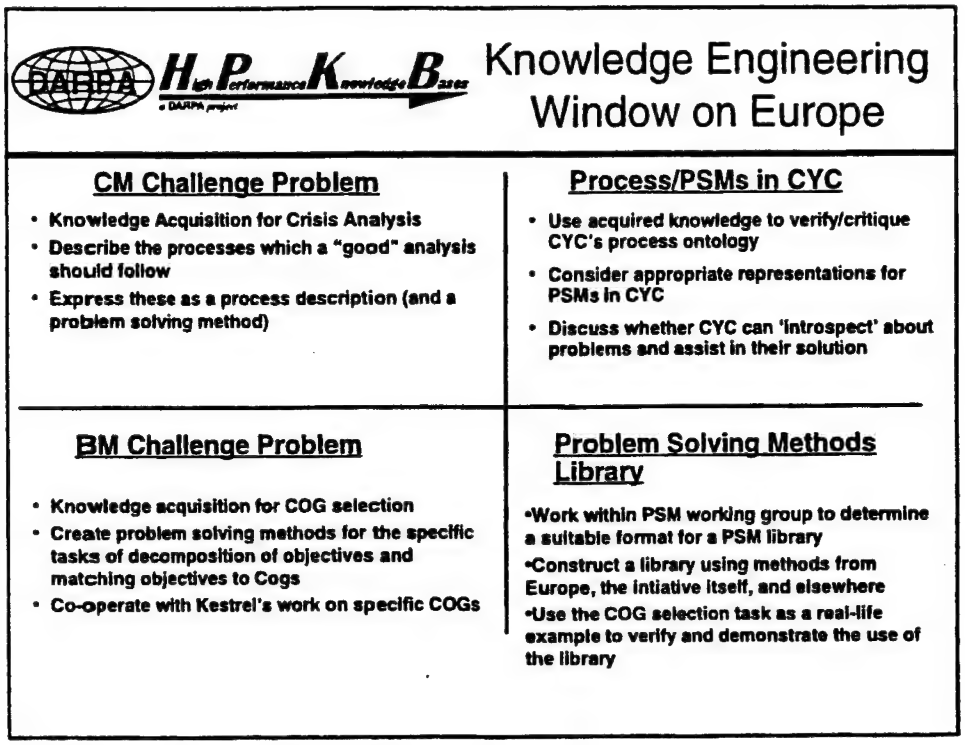
Abstract: It became apparent early in the HPKB program that AIAI could more effectively transfer European knowledge engineering practices by hands-on involvement in Challenge problem solutions rather than briefings and awareness seminars. This briefing describes AIAI's activities in acquiring knowledge for both the Crisis Analysis and Movement Analysis Challenge problems, and AIAI's desire to work with the TFS/Cycorp integration team.

Modeling approach (1)

- Model problem solving process
  - using a European top-down method
- Produce generically useful technical products
  - process & PSM for "good" crisis analysis
  - requirements and constraints on ontologies
  - process models usable within CYC
  - fully integratable PSM for objective & COG selection
- Feed back experiences to PSM working group

Modeling approach (2)

- Work with TFS integration team
  - implement CYC-compatible components
  - guide the "Make me a problem solver" task (with SMI)
  - collaborate with Kestrel on COG identification
- Contributions to demonstrators
  - process analysis as guidance or component for CM
  - COG/objective selection PSM as integrated component for BM



Knowledge Engineering Window on Europe

CM Challenge Problem

- Knowledge Acquisition for Crisis Analysis
- Describe the processes which a "good" analysis should follow
- Express these as a process description (and a problem solving method)

Process/PSMs in CYC

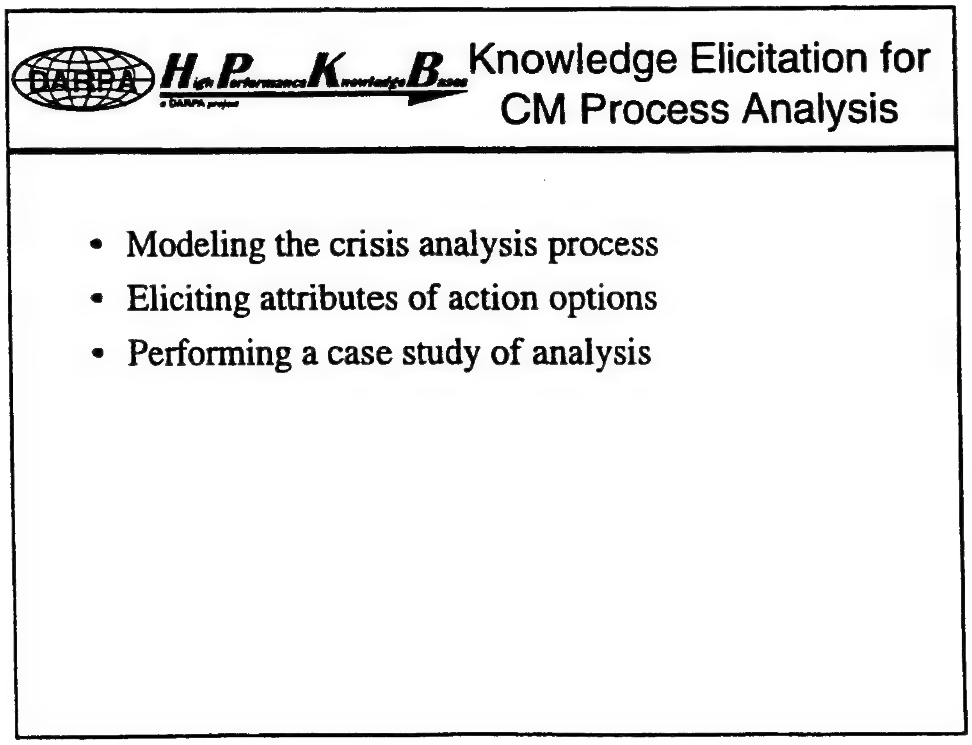
- Use acquired knowledge to verify/critique CYC's process ontology
- Consider appropriate representations for PSMs in CYC
- Discuss whether CYC can 'introspect' about problems and assist in their solution

BM Challenge Problem

- Knowledge acquisition for COG selection
- Create problem solving methods for the specific tasks of decomposition of objectives and matching objectives to Cogs
- Co-operate with Kestrel's work on specific COGs

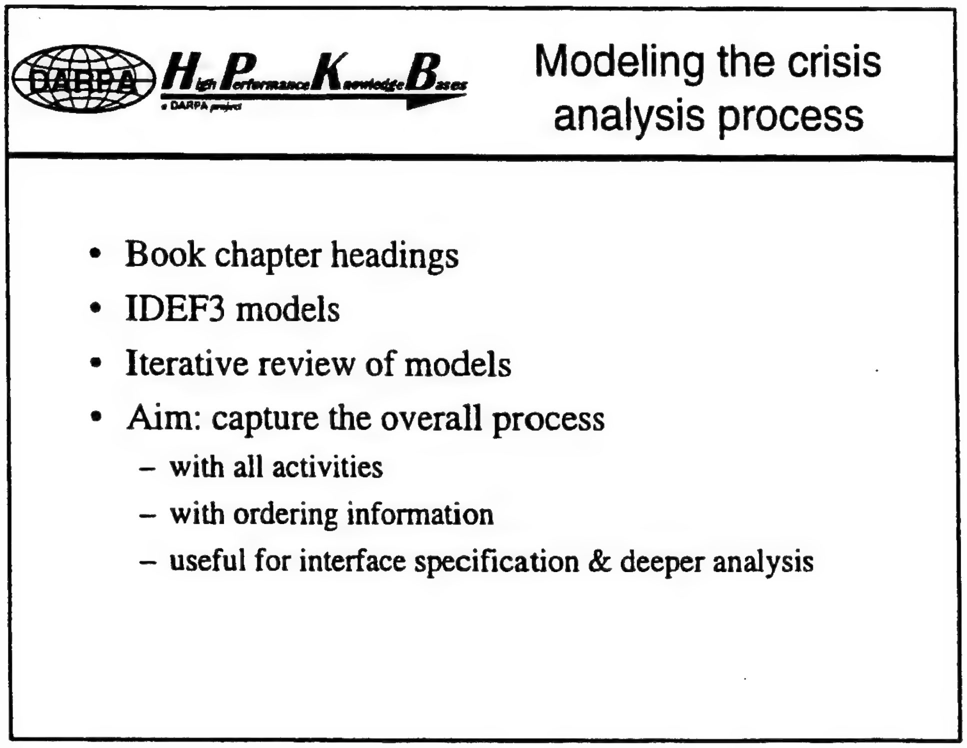
Problem Solving Methods Library

•Work within PSM working group to determine a suitable format for a PSM library
•Construct a library using methods from Europe, the intiative itself, and elsewhere
•Use the COG selection task as a real-life example to verify and demonstrate the use of the library



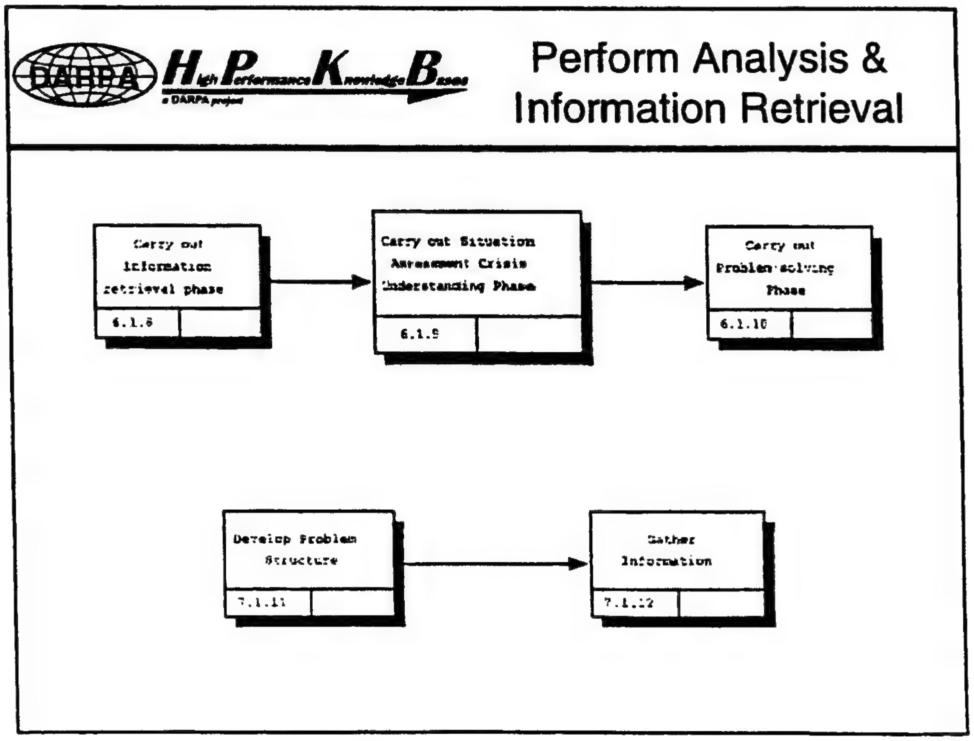
Knowledge Elicitation for CM Process Analysis

- Modeling the crisis analysis process
- Eliciting attributes of action options
- Performing a case study of analysis

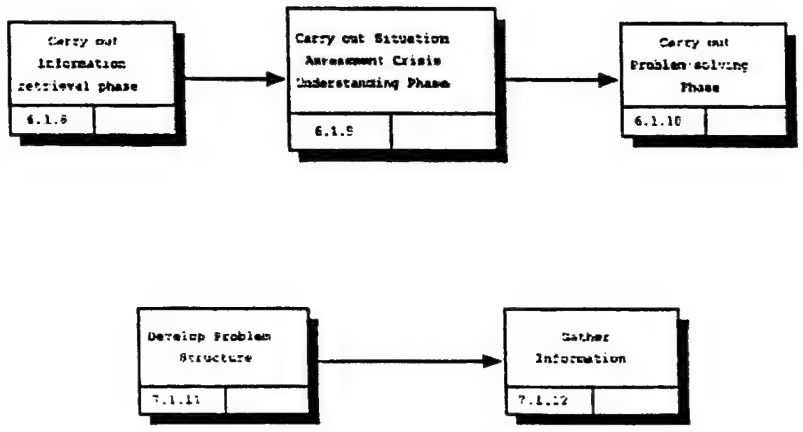


Modeling the crisis analysis process

- Book chapter headings
- IDEF3 models
- Iterative review of models
- Aim: capture the overall process
  - with all activities
  - with ordering information
  - useful for interface specification & deeper analysis

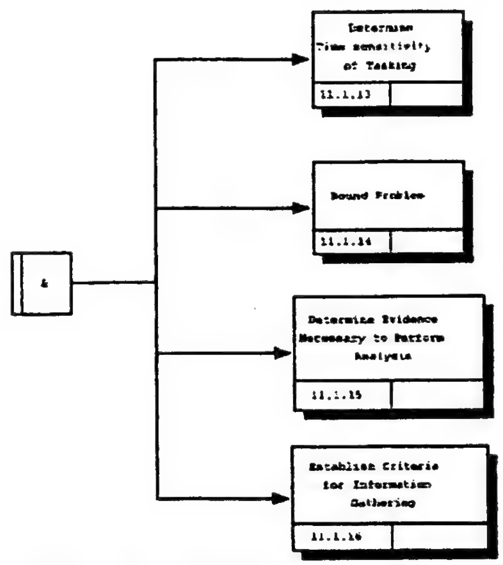


Perform Analysis & Information Retrieval


Carry out Information retrieval phase
6.1.8
Carry out Situation Assessment Crisis Understanding Phase
6.1.9
Carry out Problem-solving Phase
6.1.10
Develop Problem Structure
7.1.11
Gather Information
7.1.12

DARPA High Performance Knowledge Bases
a DARPA project

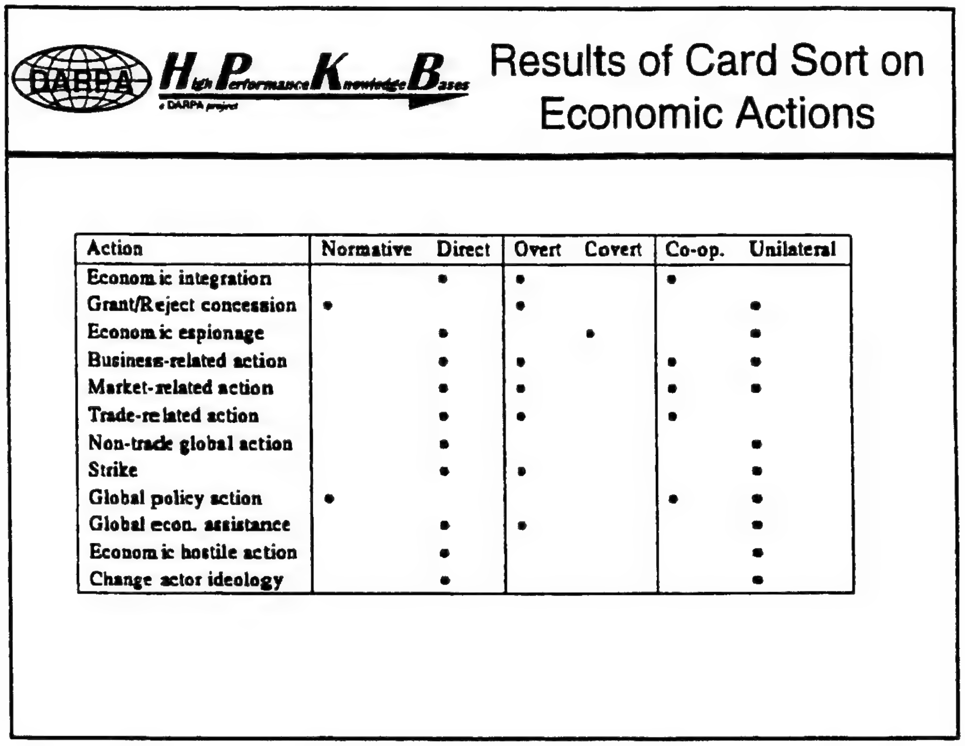
Develop Problem Structure


&
Determine Time sensitivity of Tasking
11.1.13
Bound Problem
11.1.14
Determine Evidence Necessary to Perform Analysis
11.1.15
Establish Criteria for Information Gathering
11.1.16

DARPA High Performance Knowledge Bases
a DARPA project

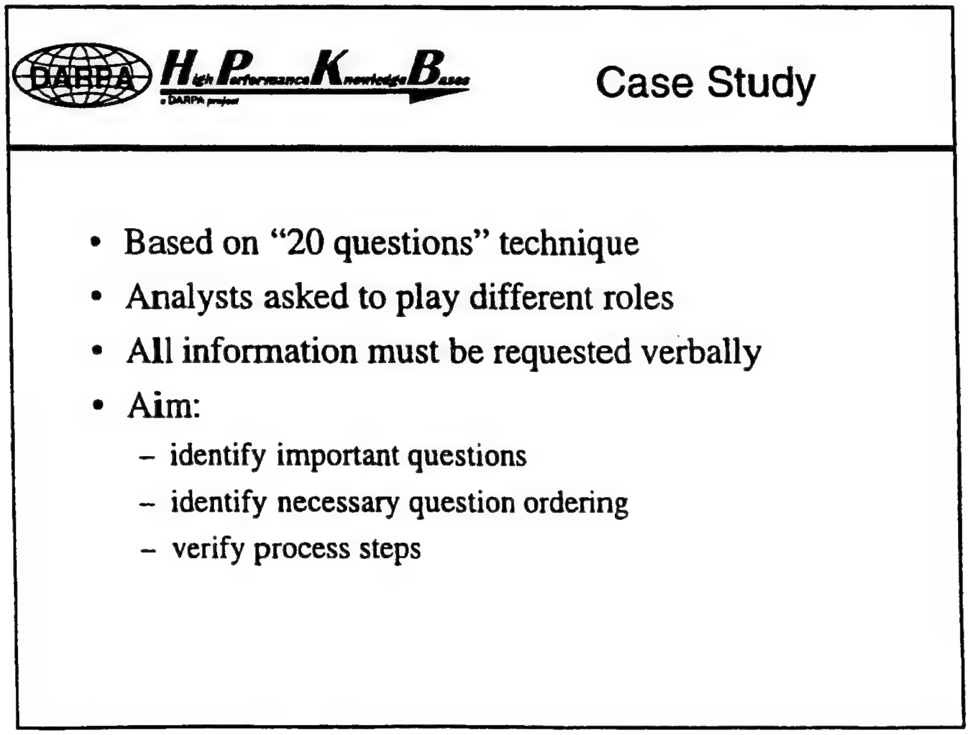
Eliciting Attributes through Card Sorting

- Action options - what each actor might do
- Write each action option on a card
- Ask an analyst to sort them in any way that seems sensible
- Repeat several times
- Aim 1: elicit distinguishing attributes
  - may be useful in filtering
- Aim 2: completeness checking
  - using semantic links



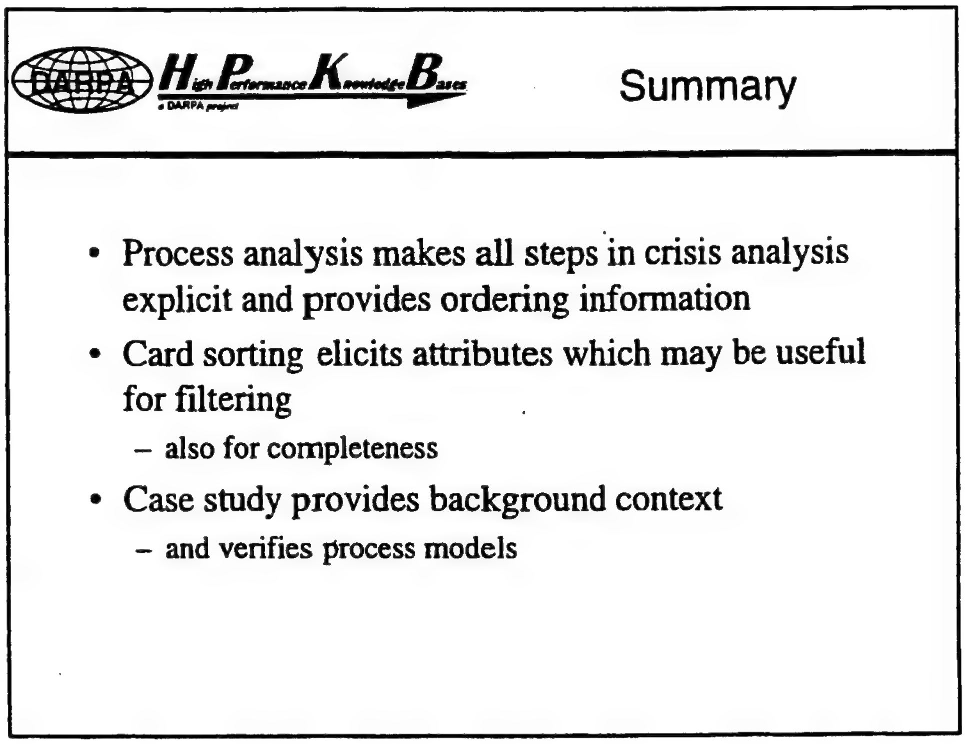
Results of Card Sort on Economic Actions

Action	Normative	Direct	Overt	Covert	Co-op.	Unilateral
Econom ic integration		•	•		•	
Grant/Reject concession	•		•			•
Econom ic espionage		•		•		•
Business-related action		•	•		•	•
Market-related action		•	•		•	•
Trade-re lated action		•	•		•	
Non-trade global action		•				•
Strike		•	•			•
Global policy action	•				•	•
Global econ. assistance		•	•			•
Econom ic hostile action		•				•
Change actor ideology		•				•



Case Study

- Based on "20 questions" technique
- Analysts asked to play different roles
- All information must be requested verbally
- Aim:
  - identify important questions
  - identify necessary question ordering
  - verify process steps



Summary

- Process analysis makes all steps in crisis analysis explicit and provides ordering information
- Card sorting elicits attributes which may be useful for filtering
  - also for completeness
- Case study provides background context
  - and verifies process models

A.3 Knowledge Engineering Window on Europe

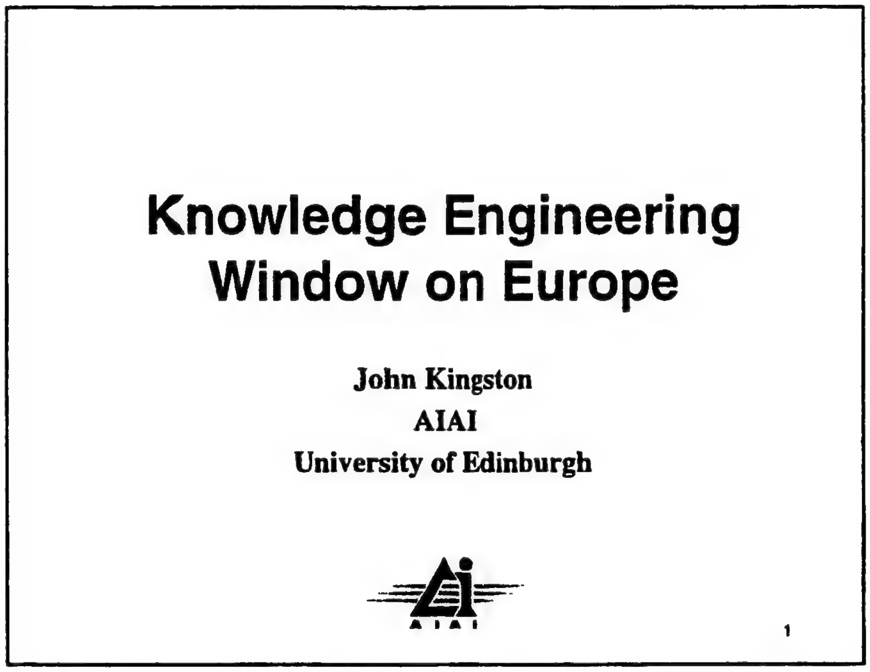
Publication details: Briefing presented at HPKB Year 1 Meeting, Washington DC, July 1998

John Kingston

Purpose: Describing AIAI's activities on the Workarounds Challenge problem

Abstract: AIAI have performed knowledge acquisition for the Workarounds Challenge problem, and have also devised a "proof of concept" solution to the Challenge problem which works entirely with Cyc's facilities for declarative theorem proving.

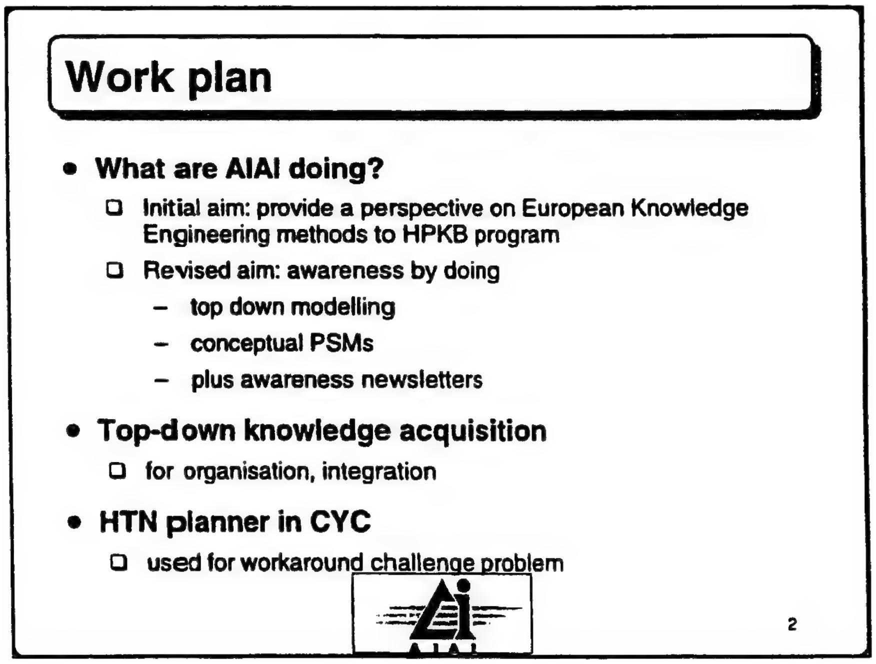
Knowledge Engineering Window on Europe - July 98 presentation



Knowledge Engineering Window on Europe

John Kingston
AIAI
University of Edinburgh

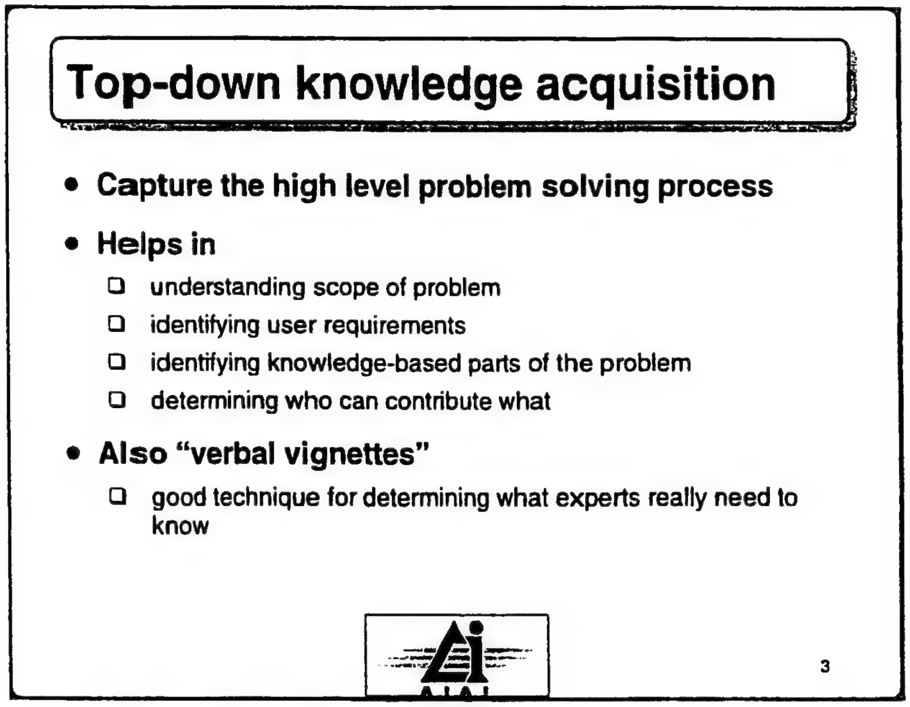
1



Work plan

- **What are AIAI doing?**
  - ❑ Initial aim: provide a perspective on European Knowledge Engineering methods to HPKB program
  - ❑ Revised aim: awareness by doing
    - top down modelling
    - conceptual PSMs
    - plus awareness newsletters
- **Top-down knowledge acquisition**
  - ❑ for organisation, integration
- **HTN planner in CYC**
  - ❑ used for workaround challenge problem

2

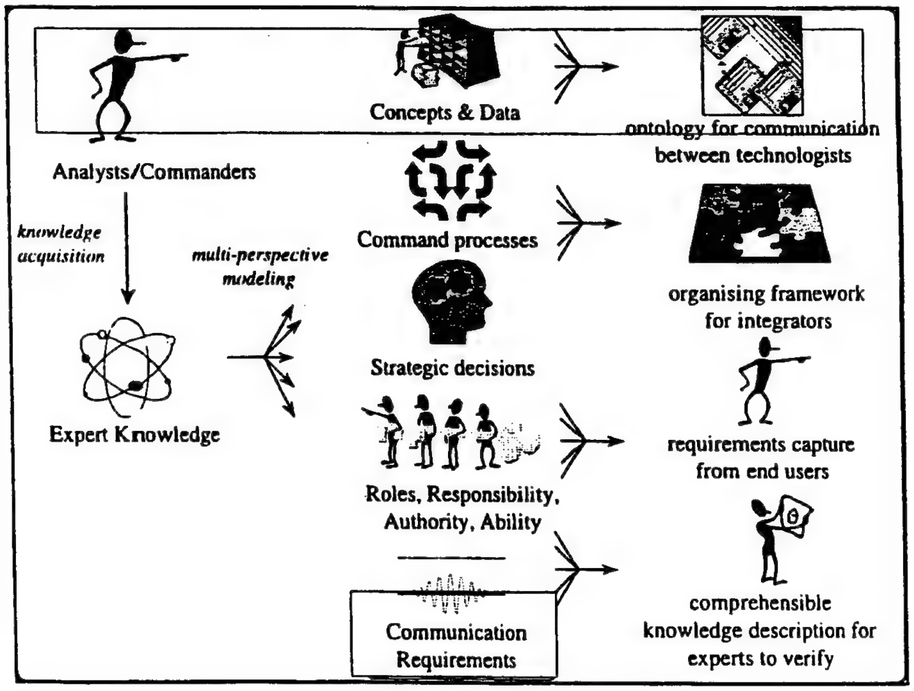


Top-down knowledge acquisition

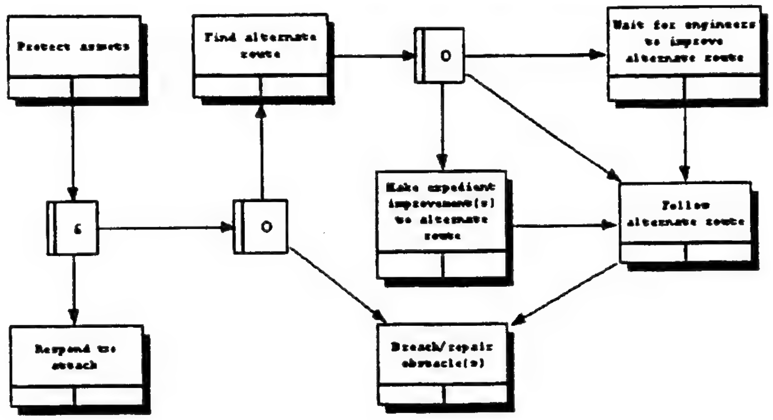
- **Capture the high level problem solving process**
- **Helps in**
  - ❑ understanding scope of problem
  - ❑ identifying user requirements
  - ❑ identifying knowledge-based parts of the problem
  - ❑ determining who can contribute what
- **Also "verbal vignettes"**
  - ❑ good technique for determining what experts really need to know

AIAI

3


Analysts/Commanders
knowledge acquisition
Expert Knowledge
multi-perspective modeling
Concepts & Data
Command processes
Strategic decisions
Roles, Responsibility, Authority, Ability
Communication Requirements
ontology for communication between technologists
organising framework for integrators
requirements capture from end users
comprehensible knowledge description for experts to verify

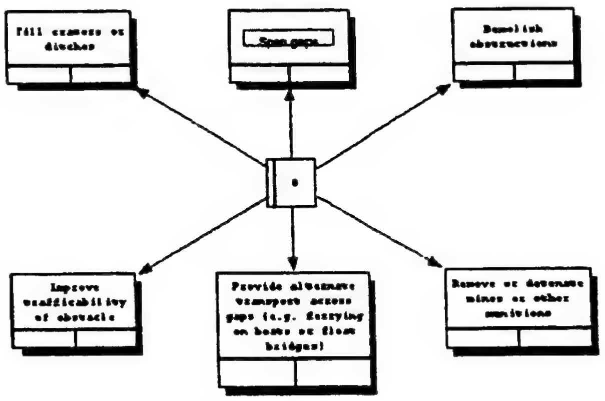
Workarounds: process diagram


Protect assets
Find alternate route
O
Wait for engineers to improve alternate route
&
O
Make expedient improvement(s) to alternate route
Follow alternate route
Respond to attack
Breach/repair obstacle(s)

AIAI

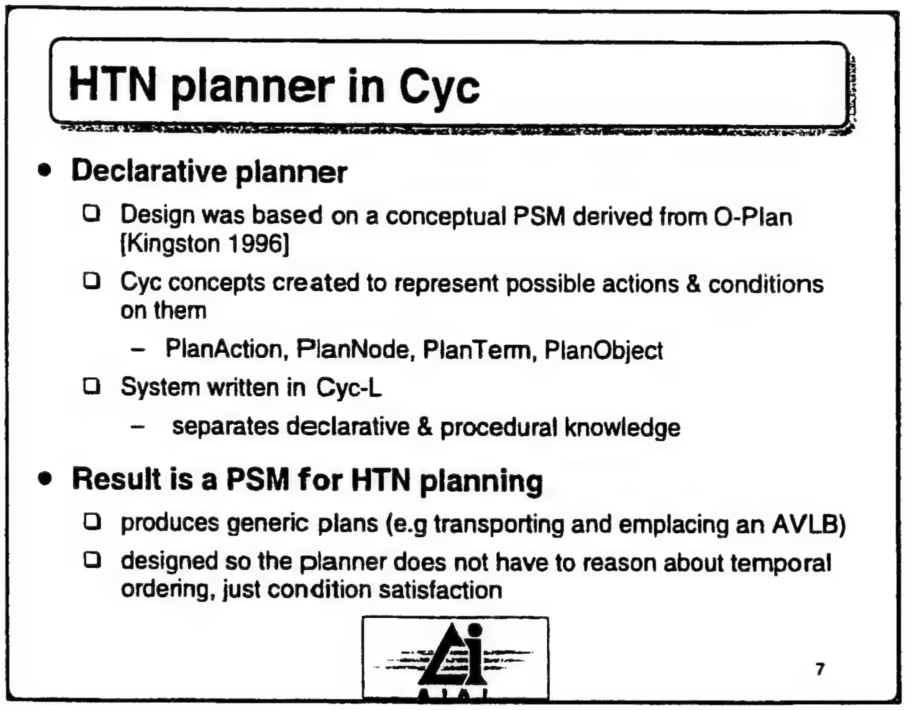
5

Workarounds: process diagram


Fill craters or ditches
Span gaps
Demolish obstructions
•
Improve trafficability of obstacle
Provide alternate transport across gaps (e.g. ferrying on boats or float bridges)
Remove or detonate mines or other munitions

AIAI

6

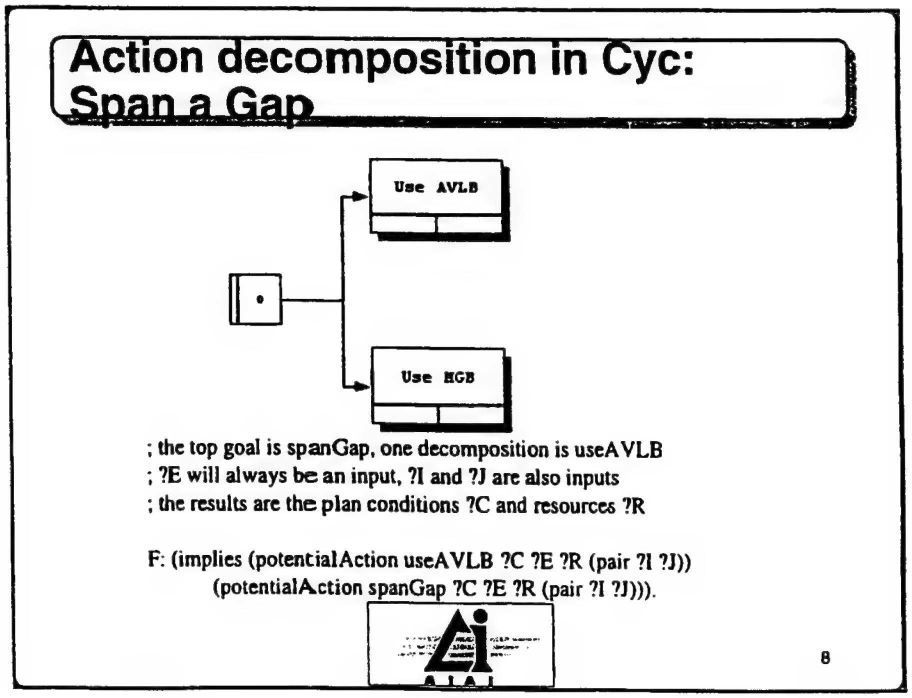


HTN planner in Cyc

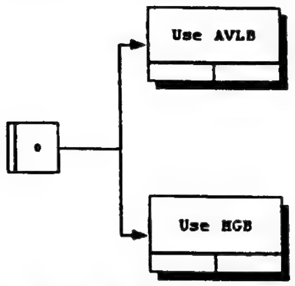
- **Declarative planner**
  - ❑ Design was based on a conceptual PSM derived from O-Plan [Kingston 1996]
  - ❑ Cyc concepts created to represent possible actions & conditions on them
    - – PlanAction, PlanNode, PlanTerm, PlanObject
  - ❑ System written in Cyc-L
    - – separates declarative & procedural knowledge
- **Result is a PSM for HTN planning**
  - ❑ produces generic plans (e.g transporting and emplacing an AVLB)
  - ❑ designed so the planner does not have to reason about temporal ordering, just condition satisfaction

AIAI

7



Action decomposition in Cyc: Span a Gap

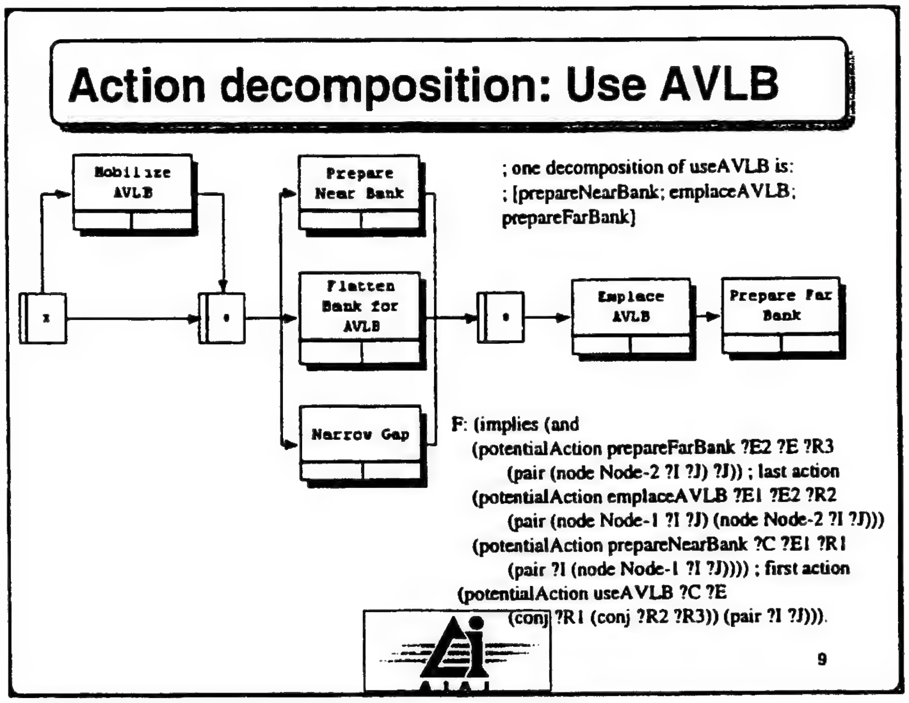


; the top goal is spanGap, one decomposition is useAVLB
; ?E will always be an input, ?I and ?J are also inputs
; the results are the plan conditions ?C and resources ?R

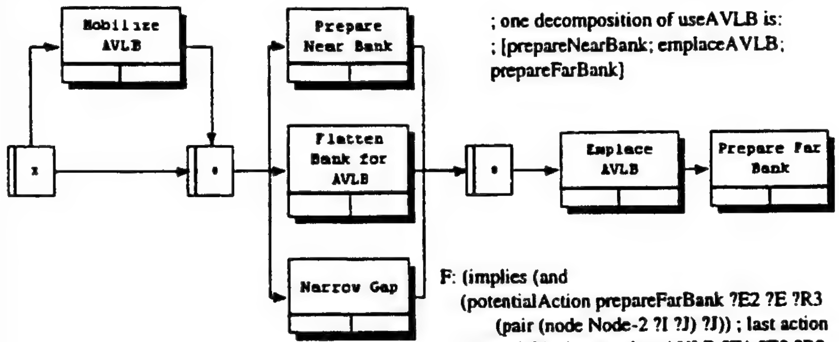
```
F: (implies (potentialAction useAVLB ?C ?E ?R (pair ?I ?J))
        (potentialAction spanGap ?C ?E ?R (pair ?I ?J))).
```

AIAI

8



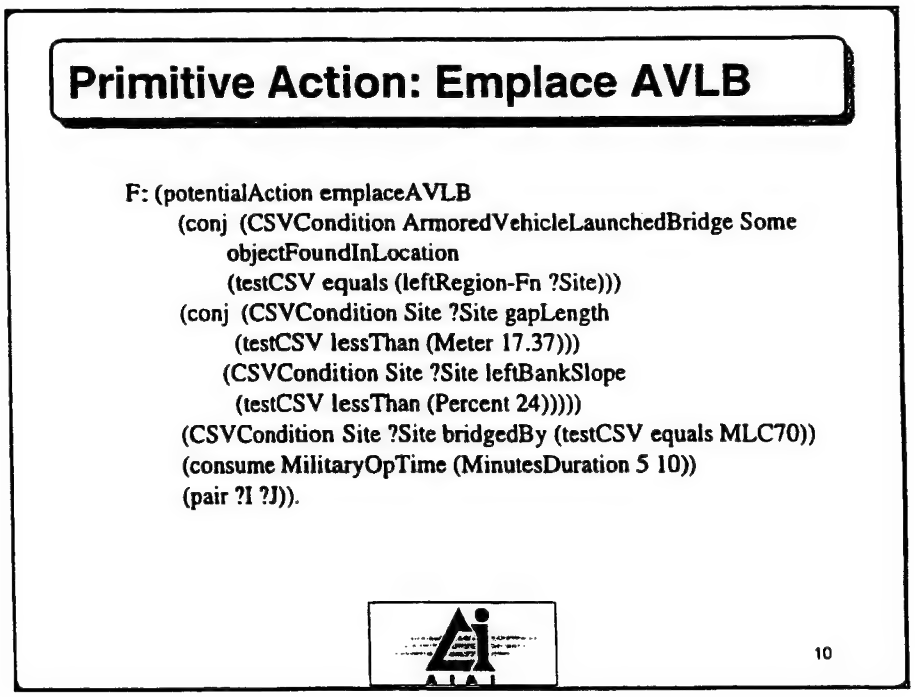
Action decomposition: Use AVLB



; one decomposition of useAVLB is:
; [prepareNearBank; emplaceAVLB;
prepareFarBank]

```
F: (implies (and
    (potentialAction prepareFarBank ?E2 ?E ?R3
        (pair (node Node-2 ?I ?J) ?J)) ; last action
    (potentialAction emplaceAVLB ?E1 ?E2 ?R2
        (pair (node Node-1 ?I ?J) (node Node-2 ?I ?J)))
    (potentialAction prepareNearBank ?C ?E1 ?R1
        (pair ?I (node Node-1 ?I ?J)))) ; first action
  (potentialAction useAVLB ?C ?E
        (conj ?R1 (conj ?R2 ?R3)) (pair ?I ?J))).
```

9

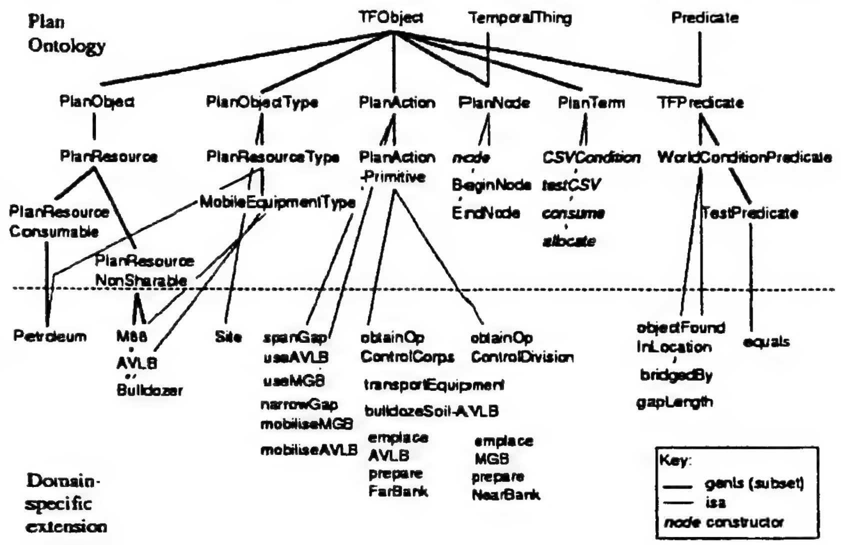


Primitive Action: Emplace AVLB

```
F: (potentialAction emplaceAVLB
     (conj (CSVCondition ArmoredVehicleLaunchedBridge Some
              objectFoundInLocation
              (testCSV equals (leftRegion-Fn ?Site)))
     (conj (CSVCondition Site ?Site gapLength
               (testCSV lessThan (Meter 17.37)))
             (CSVCondition Site ?Site leftBankSlope
               (testCSV lessThan (Percent 24)))))
     (CSVCondition Site ?Site bridgedBy (testCSV equals MLC70))
     (consume MilitaryOpTime (MinutesDuration 5 10))
     (pair ?I ?J)).
```

10

Application Ontology in Cyc



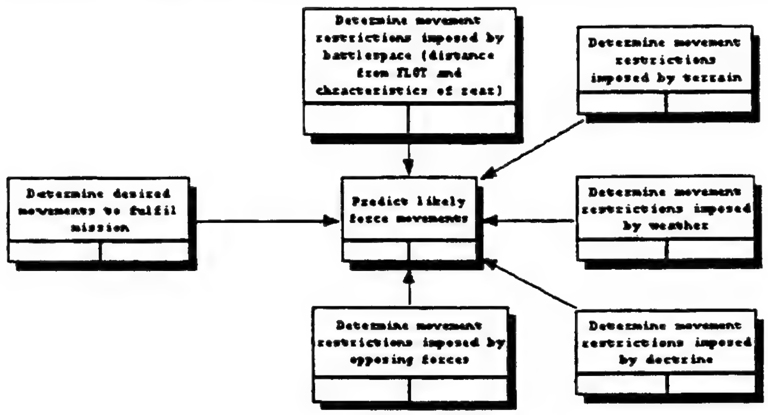
Future plans

- Extend HTN planner
- Apply the HTN planner to higher level tasks (e.g. prioritising actions within crisis management)
- Support integrators (e.g. by performing knowledge acquisition)
- Support to working groups
- Produce awareness newsletters

AIAI

12

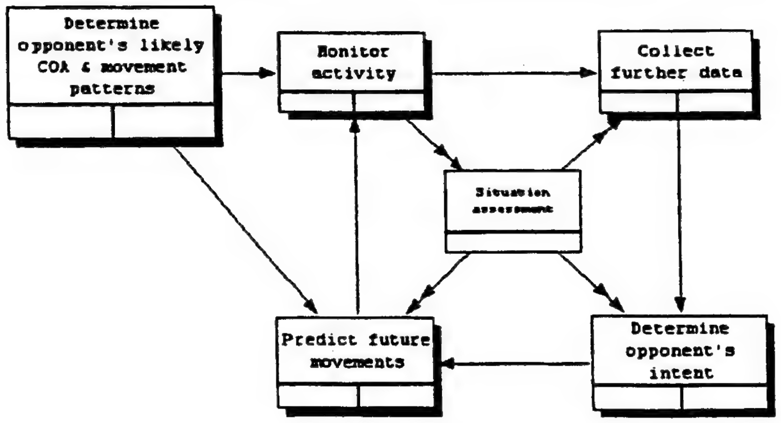
Movement analysis: process diagram


Determine movement restrictions imposed by battlespace (distance from FLOT and characteristics of rear)
Determine movement restrictions imposed by terrain
Determine desired movements to fulfil mission
Predict likely force movements
Determine movement restrictions imposed by weather
Determine movement restrictions imposed by opposing forces
Determine movement restrictions imposed by doctrine

A I A I

13

Movement Analysis: process diagrams



A I A I

14

A.4 Getting There from Here

Publication details: Briefing presented at HPKB Year 1.5 Meeting, Austin, Texas, January 1999

Stuart Aitken and John Kingston

Purpose: To describe AIAI's proposed work on the COA Challenge problem

Abstract: AIAI took on the task of writing a parser in Prolog to convert statements in a structured grammar into machine-readable statements, and to "fuse" these with statements emanating from a sketch tool. This briefing describes the proposed approach.

Getting there from here

Stuart Aitken and John Kingston
AIAI
Division of Informatics
University of Edinburgh

1

Here and there

- **Starting point:**
  - COA statement
  - COA sketch
- **End goal:**
  - knowledge representation suitable for critiquing
- **How:**
  - Parsing
  - Fusion
  - Knowledge Acquisition

Parsing

Prolog parser to convert COA statement to axioms

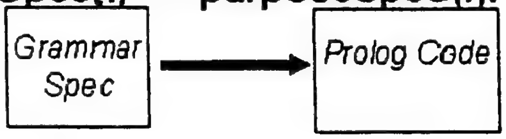
A parser generator was written:

- Input: the formal grammar spec:

<TaskSpec> = [{...}]...[<PurposeSpec>]...

- Output: Prolog grammar rules

taskSpec(I) --> purposeSpec(I).



Parsing and Interpretation

Parser code modified and extended

Interpretation by writing functions to descend the parse tree

- and to provide output in either KE-text or CycL format

Unify assertions - merging with sketch tool output

- some inference needed

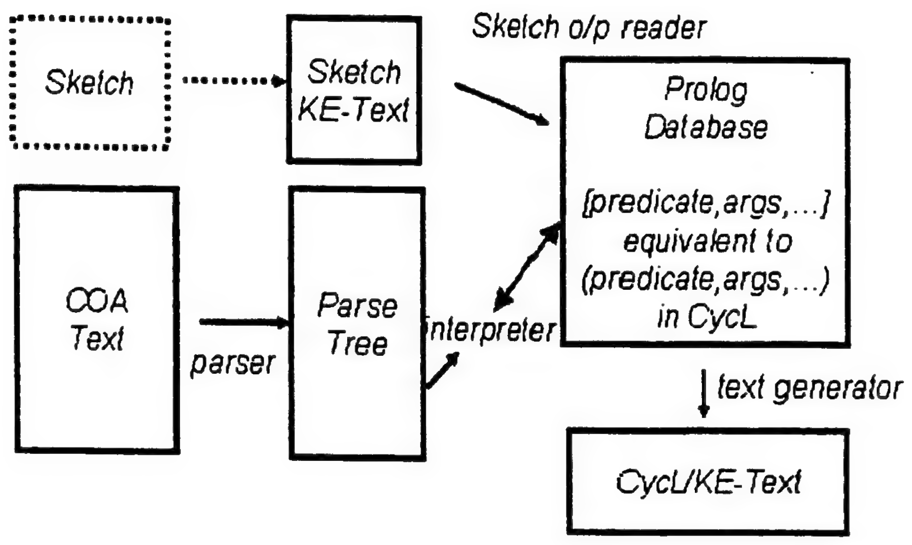
Metrics

- Parser generator: 82 clauses
- Grammar rules generated: 993
- Grammar rules written: 167
- Interpreter & other: 253 clauses
- Total time: 11.5 man days

5

Fusion

- Fusion merges the interpreted COA text with the Prolog database derived from the sketch tool output
- Without the sketch input, the interpreter generates names for units and tasks, constructs assertions



Knowledge Acquisition

- Some CP questions require further K.A.
- Those which require ontology development:
  - Is the unit capable of performing its assigned task?
  - Does this task accomplish its stated purpose?
- Unit ontology
  - combat/combat support/combat service support
  - mobile/transportable/static
  - land/airborne
- Task ontology
- Purpose ontology

Extend parsing

- temporal, spatial information

Pre-critiquing

- parser/interpreter could perform completeness critiquing

Further knowledge acquisition needed

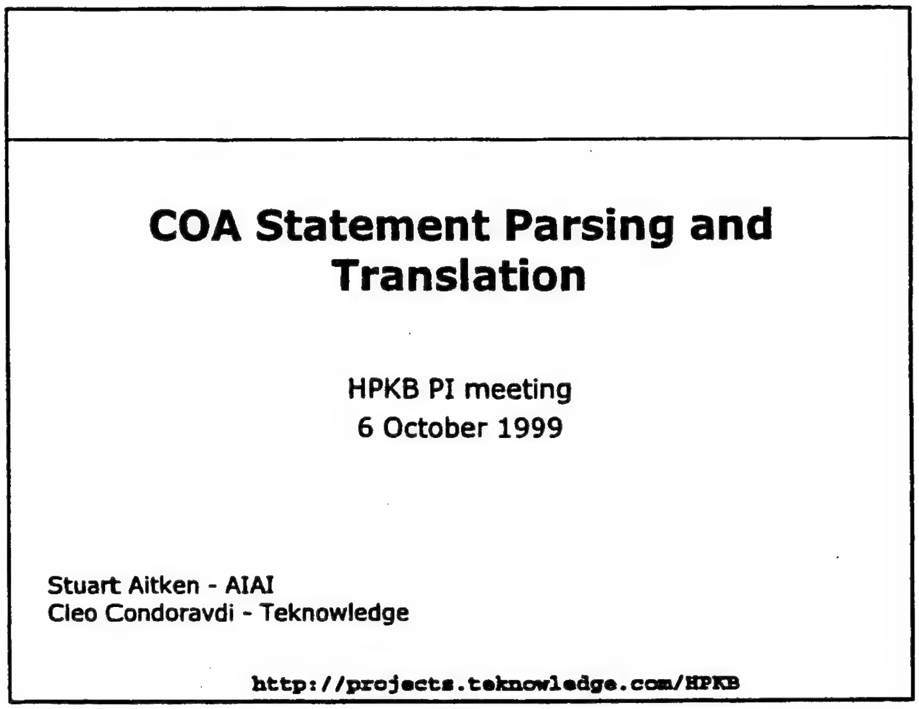
A.5 COA Statement Parsing and Translation

Publication details: Briefing presented at HPKB Final Meeting, Washington DC, October 1999

Stuart Aitken and Cleo Condoravdi

Purpose: To describe the work on parsing and fusion of COA statements carried out by AIAI in conjunction with Teknowledge

Abstract: AIAI worked with Teknowledge and Cycorp to produce a statement translator for use in the COA Challenge problem. This brief outlines its design and implementation.

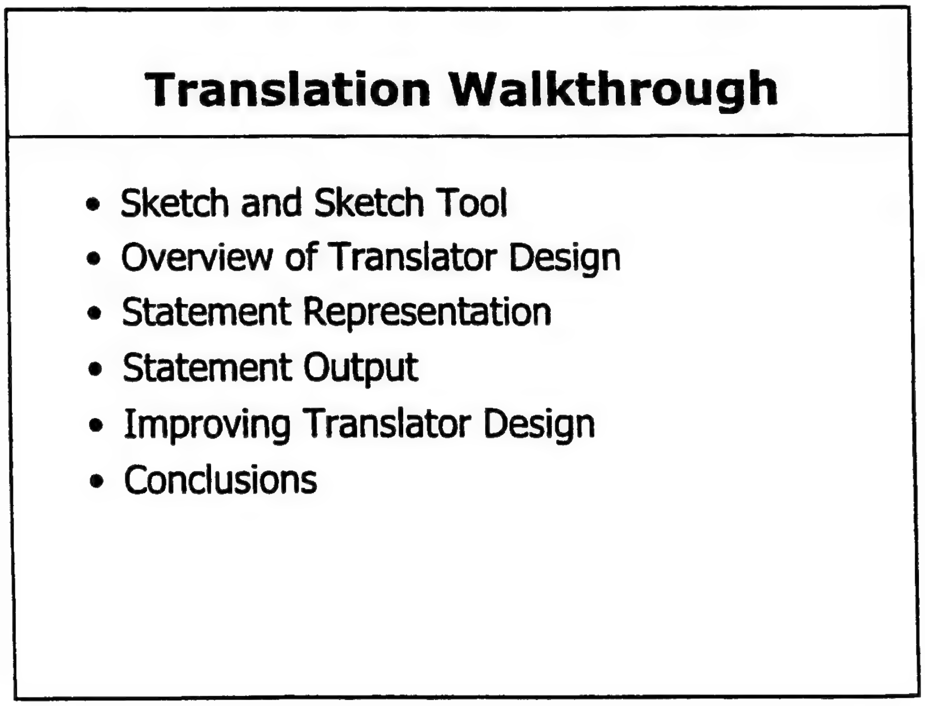


COA Statement Parsing and Translation

HPKB PI meeting
6 October 1999

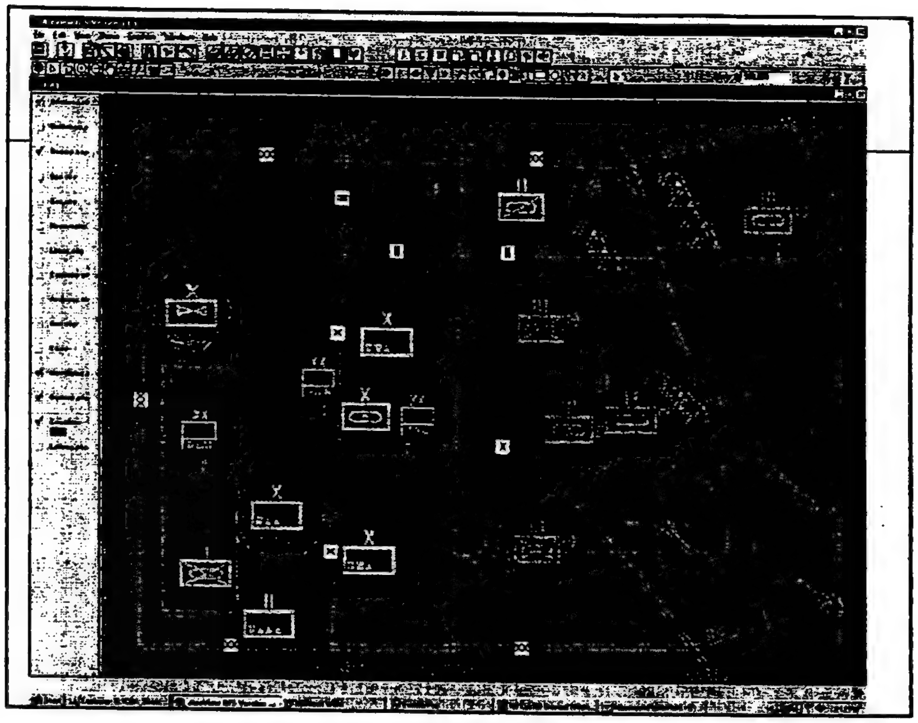
Stuart Aitken - AIAI
Cleo Condoravdi - Teknowledge

http://projects.teknowledge.com/HPKB



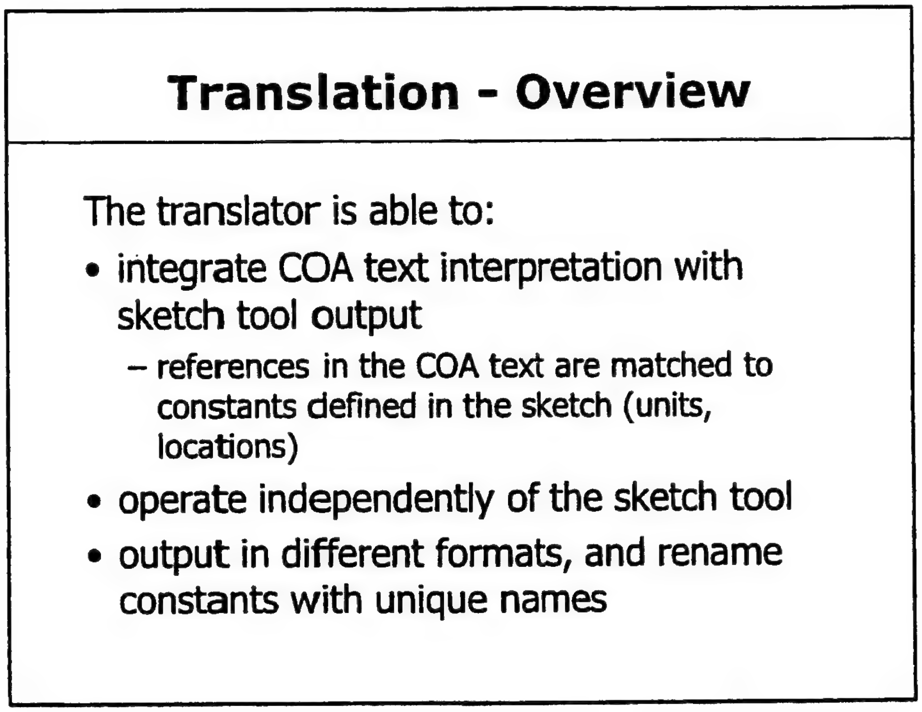
Translation Walkthrough

- Sketch and Sketch Tool
- Overview of Translator Design
- Statement Representation
- Statement Output
- Improving Translator Design
- Conclusions



Sketch Tool

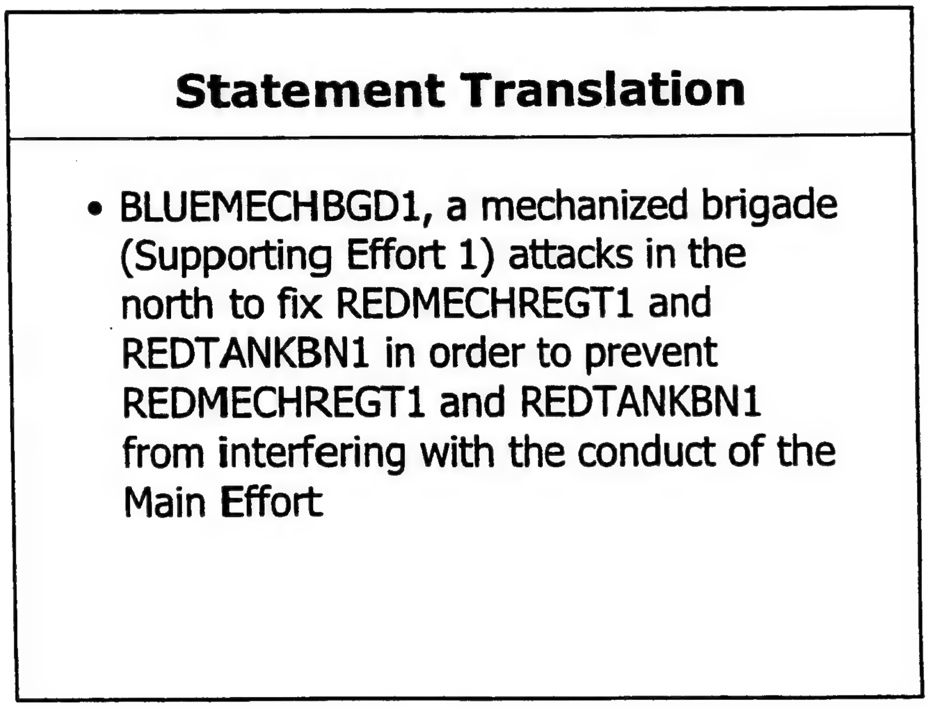
- Generates instances of units and labels naming them
- Generates descriptions of boundary and phase lines, of areas defined by control measures, and of important locations e.g. OBJ SLAM and MUDDY RIVER
- Defines those tasks with a corresponding visual symbol



Translation - Overview

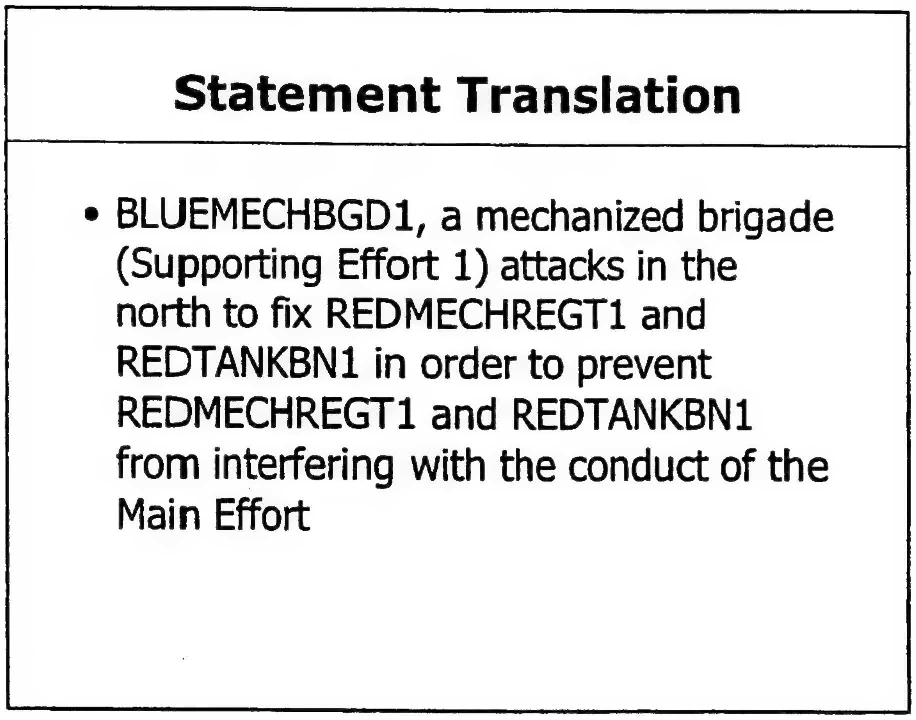
The translator is able to:

- integrate COA text interpretation with sketch tool output
  - references in the COA text are matched to constants defined in the sketch (units, locations)
- operate independently of the sketch tool
- output in different formats, and rename constants with unique names



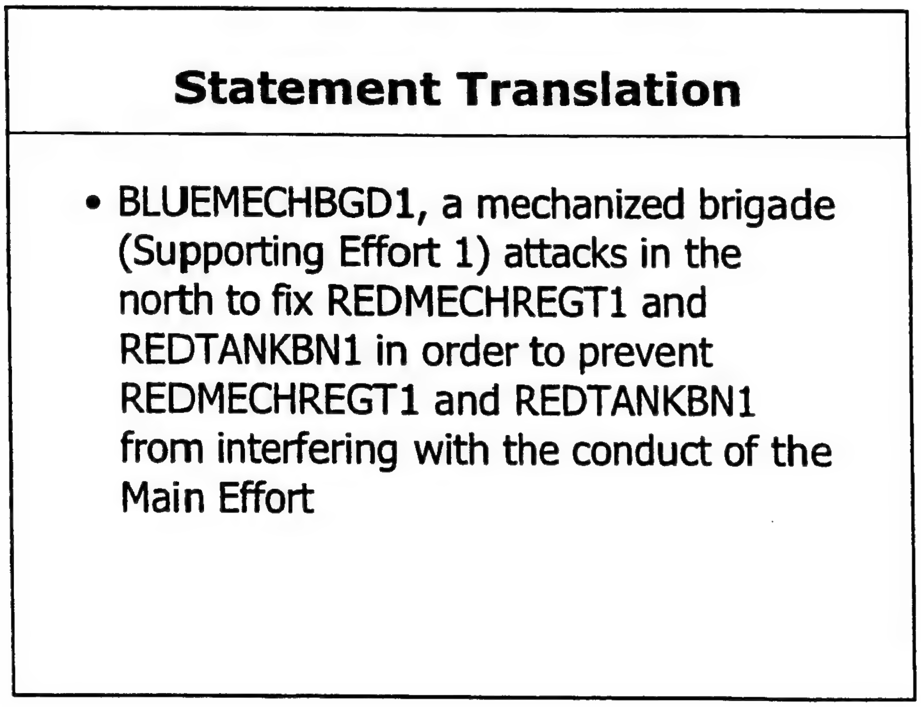
Statement Translation

- BLUEMECHBGD1, a mechanized brigade (Supporting Effort 1) attacks in the north to fix REDMECHREGT1 and REDTANKBN1 in order to prevent REDMECHREGT1 and REDTANKBN1 from interfering with the conduct of the Main Effort



Statement Translation

- BLUEMECHBGD1, a mechanized brigade (Supporting Effort 1) attacks in the north to fix REDMECHREGT1 and REDTANKBN1 in order to prevent REDMECHREGT1 and REDTANKBN1 from interfering with the conduct of the Main Effort



Statement Translation

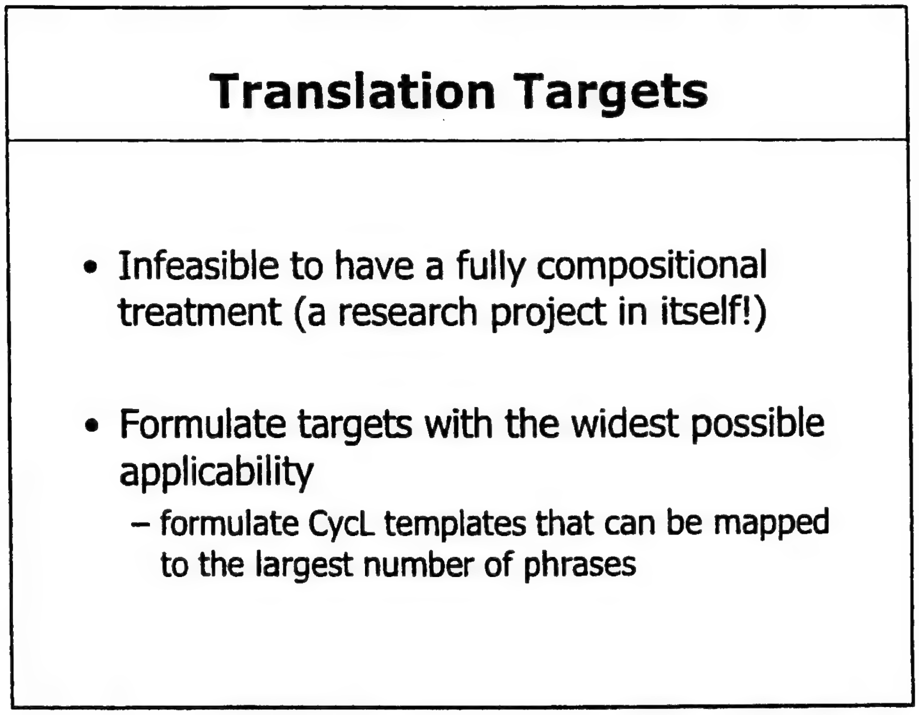
- BLUEMECHBGD1, a mechanized brigade (Supporting Effort 1) attacks in the north to fix REDMECHREGT1 and REDTANKBN1 in order to prevent REDMECHREGT1 and REDTANKBN1 from interfering with the conduct of the Main Effort

Statement Translation

- BLUEMECHBGD1, a mechanized brigade (Supporting Effort 1) attacks in the north to fix REDMECHREGT1 and REDTANKBN1 in order to prevent REDMECHREGT1 and REDTANKBN1 from interfering with the conduct of the Main Effort

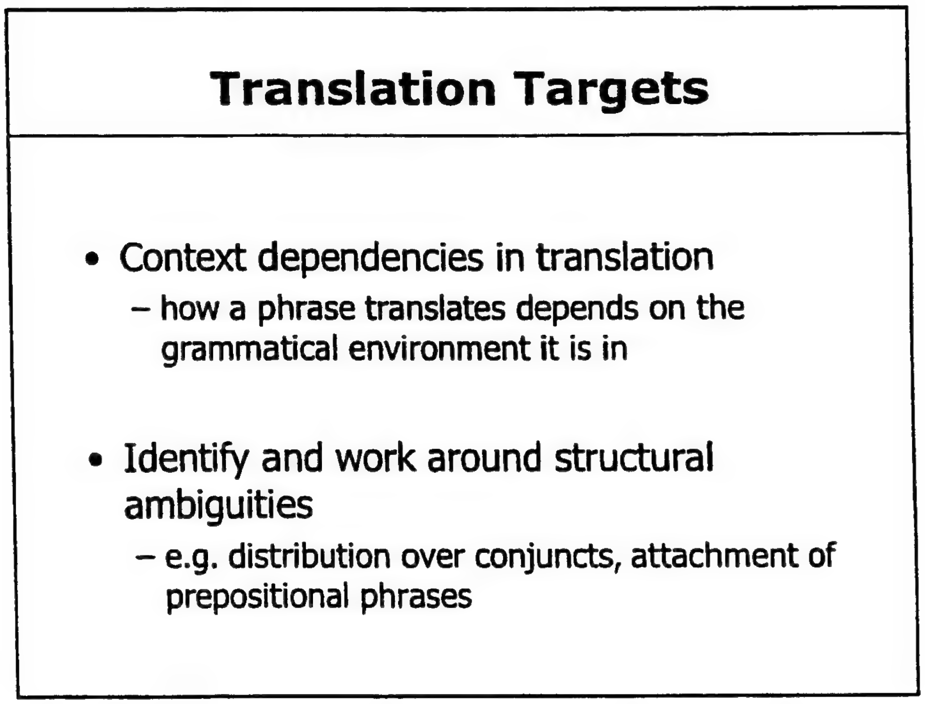
Translation- Approach

- The input text is parsed sentence by sentence
- Each resulting parse tree is interpreted (first pass) to construct CycL assertions and asserting facts in a database
- Selected parse trees are interpreted a second time in the context of the CycL and database assertions e.g. to resolve references in purpose statements to tasks performed by other units



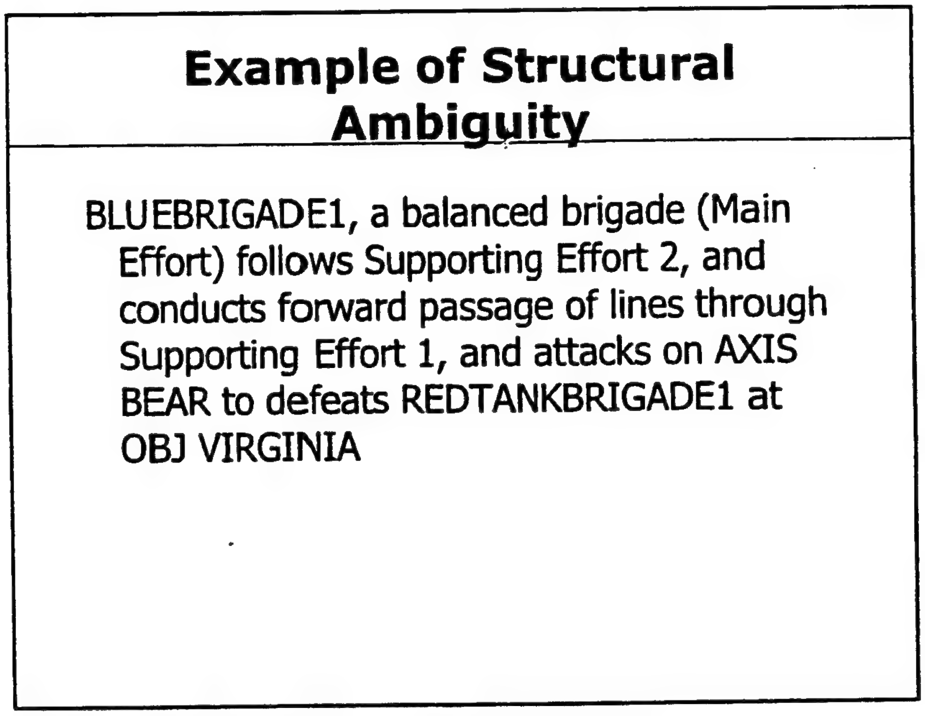
Translation Targets

- Infeasible to have a fully compositional treatment (a research project in itself!)
- Formulate targets with the widest possible applicability
  - formulate CycL templates that can be mapped to the largest number of phrases



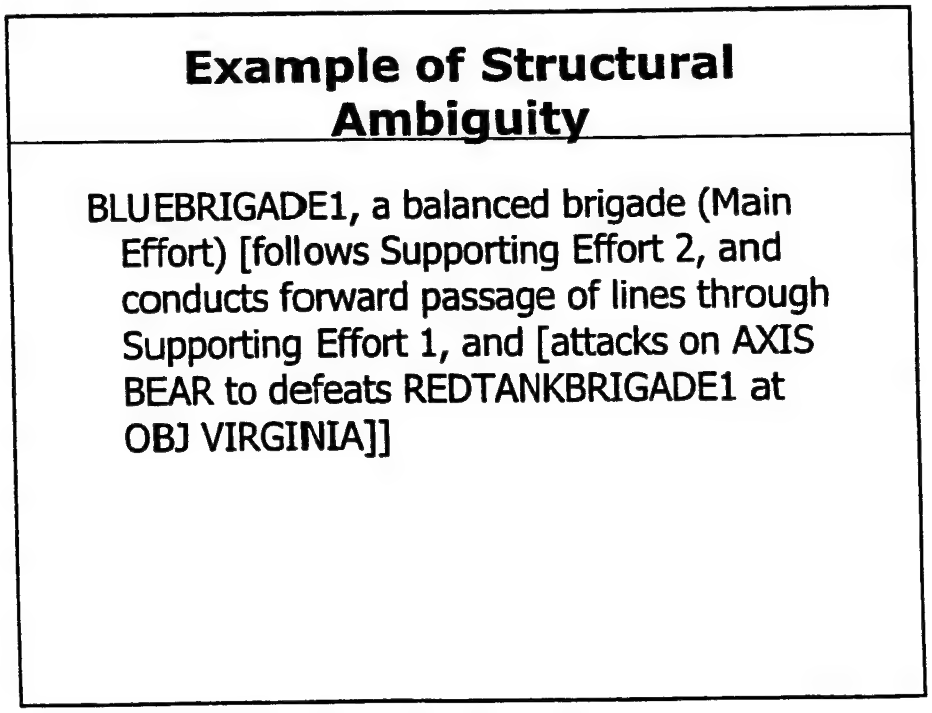
Translation Targets

- Context dependencies in translation
  - how a phrase translates depends on the grammatical environment it is in
- Identify and work around structural ambiguities
  - e.g. distribution over conjuncts, attachment of prepositional phrases



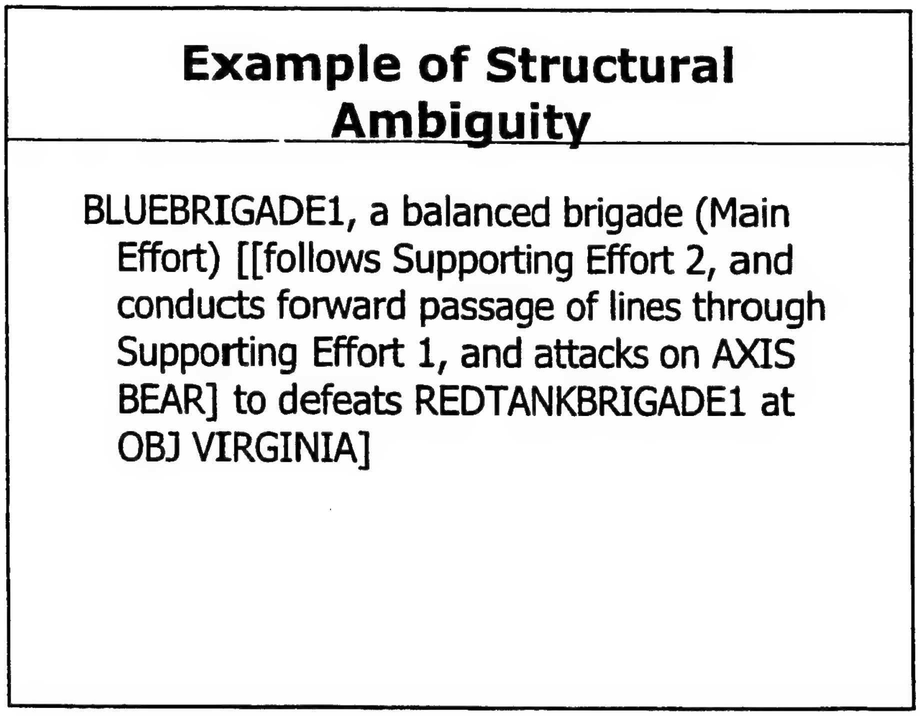
Example of Structural Ambiguity

BLUEBRIGADE1, a balanced brigade (Main Effort) follows Supporting Effort 2, and conducts forward passage of lines through Supporting Effort 1, and attacks on AXIS BEAR to defeats REDTANKBRIGADE1 at OBJ VIRGINIA



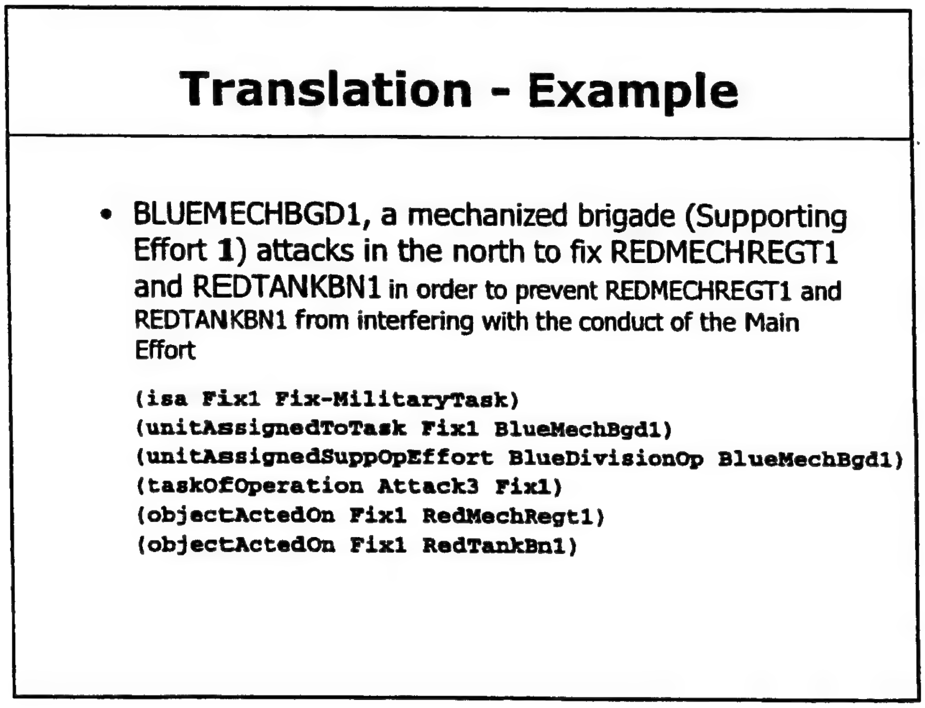
Example of Structural Ambiguity

BLUEBRIGADE1, a balanced brigade (Main Effort) [follows Supporting Effort 2, and conducts forward passage of lines through Supporting Effort 1, and [attacks on AXIS BEAR to defeats REDTANKBRIGADE1 at OBJ VIRGINIA]]



Example of Structural Ambiguity

BLUEBRIGADE1, a balanced brigade (Main Effort) [[follows Supporting Effort 2, and conducts forward passage of lines through Supporting Effort 1, and attacks on AXIS BEAR] to defeats REDTANKBRIGADE1 at OBJ VIRGINIA]



Translation - Example

- BLUEMECHBGD1, a mechanized brigade (Supporting Effort **1**) attacks in the north to fix REDMECHREGT1 and REDTANKBN1 in order to prevent REDMECHREGT1 and REDTANKBN1 from interfering with the conduct of the Main Effort

```
(isa Fix1 Fix-MilitaryTask)
(unitAssignedToTask Fix1 BlueMechBgd1)
(unitAssignedSuppOpEffort BlueDivisionOp BlueMechBgd1)
(taskOfOperation Attack3 Fix1)
(objectActedOn Fix1 RedMechRegt1)
(objectActedOn Fix1 RedTankBn1)
```

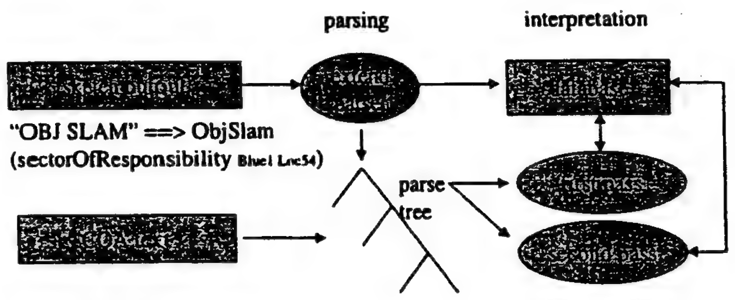
Translation - Example

- BLUEMECHBGD1, a mechanized brigade (Supporting Effort 1) attacks in the north to fix REDMECHREGT1 and REDTANKBN1 in order to prevent REDMECHREGT1 and REDTANKBN1 from interfering with the conduct of the Main Effort

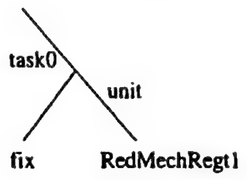
```
(taskHasPurpose Fix1
 (prevents-SitSitType Fix1
    (CollectionSubsetFn MilitaryInterferenceAction
      (TheSetOf ?OBJ
         (and (performedBy ?OBJ RedMechRegt1)
                (objectActedOn ?OBJ BlueTankBgd1))))))
```

Note how "fix" is represented as an instance, while "interfering" is a type level expression

Translation - Process



Translation - Rules



Input pattern:

[task0 fix [unit RedMechRegt1]]

Matching pattern:

[task0 T [unit0 R]]   Agent=U

(unitAssignedToTask T' U) (objectActedOn T' R)

{T' = generateTask(T,U)}

Translation

Problems:

- information relevant for pattern matching may be distributed across a large parse tree
  - extract relevant subtrees and use these in matching
- token names and sketch information are not in the parse tree at all
  - need to pass parameters into the interpretation algorithm

Translation - Rules

- Subtrees of the original parse tree are extracted, enabling pattern matching on the simplified term

E.g. the original parser output:

```
[[unit [COANamedUnit
  ...[...]]][typeOfOperationStatement[...]
   [task0 fix [unit ...]] [purpose [...]]
```

is reduced to

```
[task0 fix [unit RedMechRegt1]]
```

Translation - Rules

Input pattern:

```
[task0 fix [unit RedMechRegt1]]
```

Matching pattern:

```
[task0 T [unit R]]     Agent=U
(unitAssignedToTask T' U) (P T' R)
{T' = generateTask(T,U,R)
  P = selectRelation(T,unit)
  T=/=moves}
```

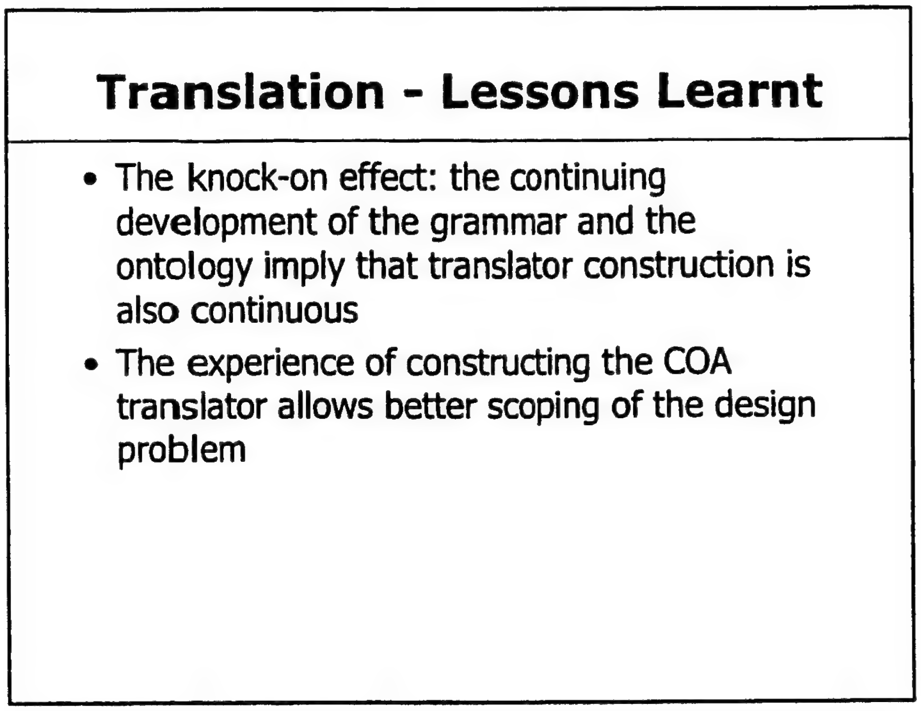
Translation - Process

Interpretation

- 1st pass: parse each sentence, extract subtrees, pattern match on each subtree
  - 12 types of subtrees
  - up to 30 interpretation rules per subtree
- 2nd pass: further resolve references
  "enable the completion of"
  (taskHasPurpose Fix1
      (enables-EventEvent Fix1 Seize1))
- final step: unify constant names

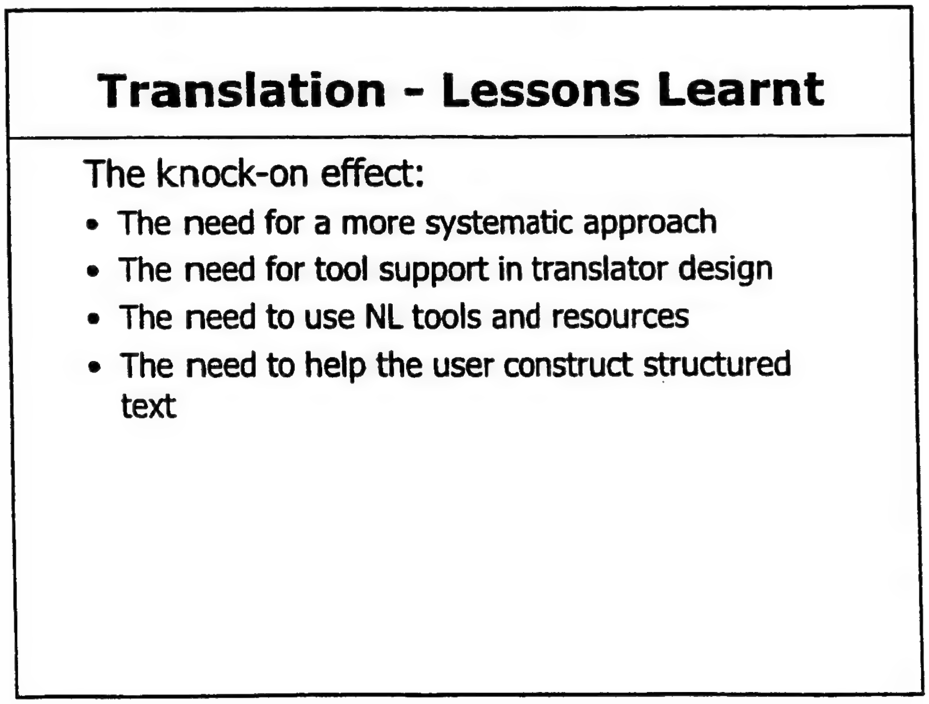
Contextual Factors

- Resolving references
  - to expressions appearing elsewhere in COA
    e.g. "BlueBrigade1 (the Main Effort) ..."
    " .... the Main Effort ... "
  - to elements appearing in the sketch e.g.
    "area of operations of UnitX"
    resolve using (sectorOfResponsibility)
  - to elements of the formal representation e.g
    <the main task of operation>
    <the operation name>



Translation - Lessons Learnt

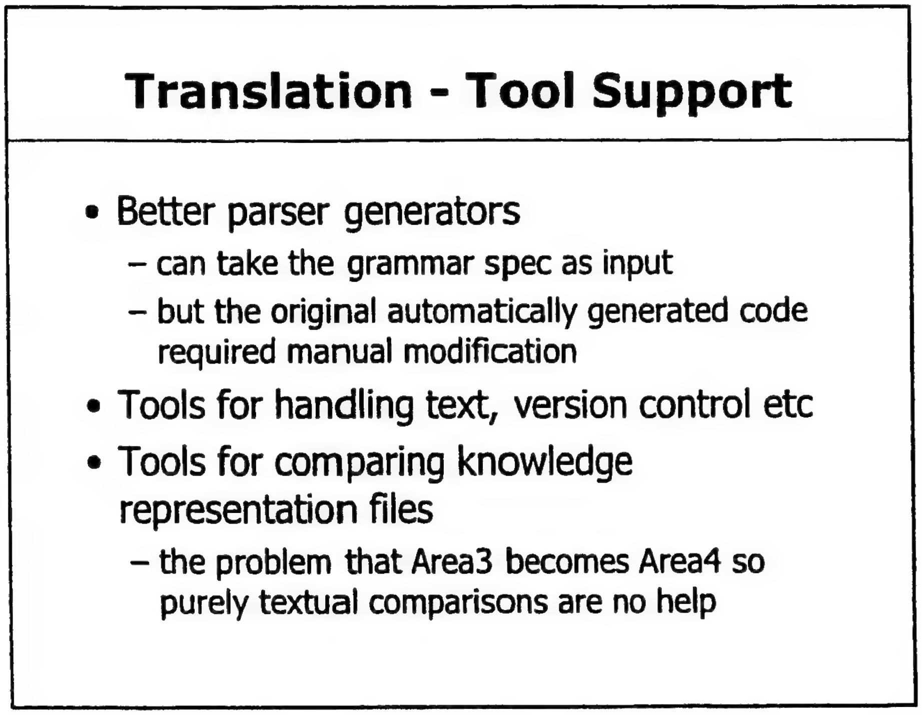
- The knock-on effect: the continuing development of the grammar and the ontology imply that translator construction is also continuous
- The experience of constructing the COA translator allows better scoping of the design problem



Translation - Lessons Learnt

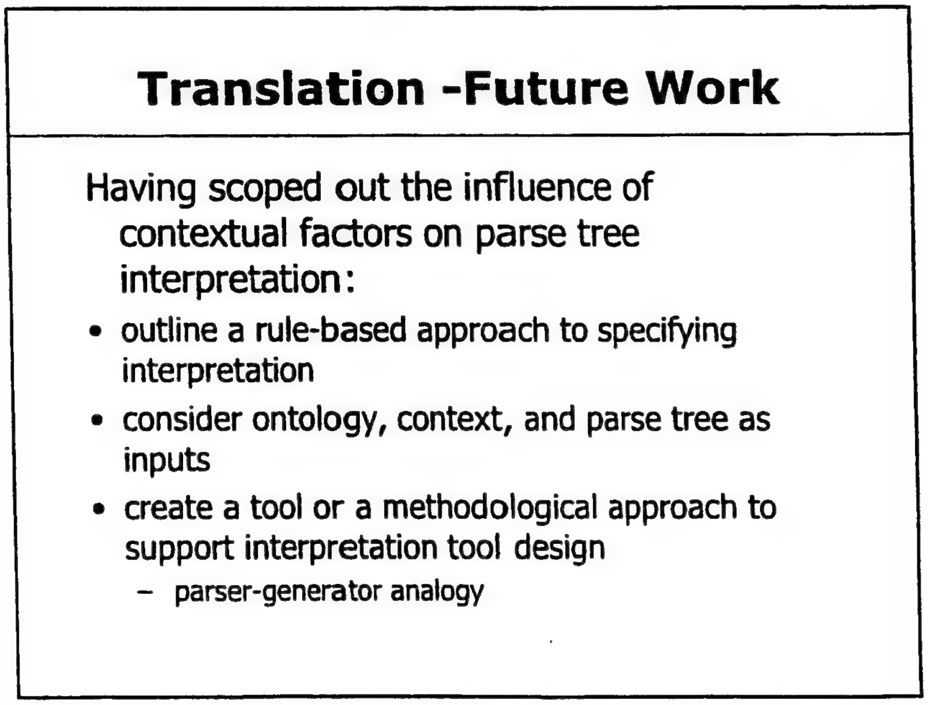
The knock-on effect:

- The need for a more systematic approach
- The need for tool support in translator design
- The need to use NL tools and resources
- The need to help the user construct structured text



Translation - Tool Support

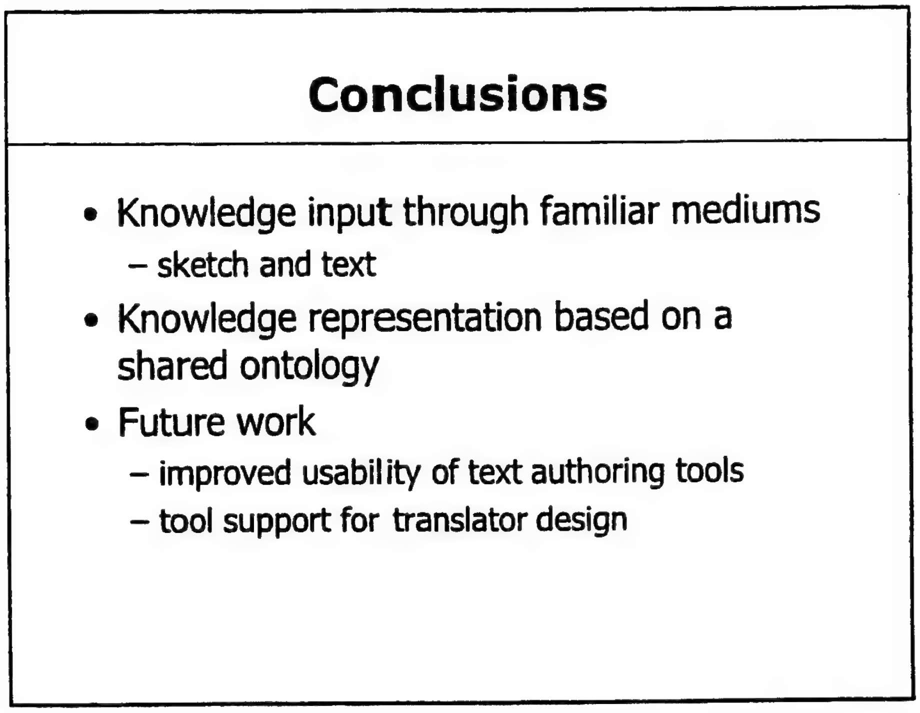
- Better parser generators
  - can take the grammar spec as input
  - but the original automatically generated code required manual modification
- Tools for handling text, version control etc
- Tools for comparing knowledge representation files
  - the problem that Area3 becomes Area4 so purely textual comparisons are no help



Translation -Future Work

Having scoped out the influence of contextual factors on parse tree interpretation:

- outline a rule-based approach to specifying interpretation
- consider ontology, context, and parse tree as inputs
- create a tool or a methodological approach to support interpretation tool design
  - parser-generator analogy



Conclusions

- Knowledge input through familiar mediums
  - sketch and text
- Knowledge representation based on a shared ontology
- Future work
  - improved usability of text authoring tools
  - tool support for translator design

B Newsletters

This appendix contains deliverables N-1, N-2 and N-3, which were the two newsletters prepared as part of AIAI's original role in making HPKB participants aware of activities in Europe. This work package was scaled down after the first newsletter; the second newsletter was preapred because of interest expressed by some HPKB participants.

Ai

High Performance Knowledge Bases
a DARPA project

Editorial

This is the first of a series of newsletters which AIAI will produce for HPKB participants. The aim of these newsletters is to inform HPKB participants about relevant academic work in Europe and elsewhere. The first issue contains a report on a workshop held at IJCAI-97 entitled "Problem-Solving Methods for Knowledge-Based Systems". The workshop organisers were Dieter Fensel of DFE, Karlsruhe University, Germany, and Richard Benjamins of SWI, University of Amsterdam, Netherlands;. Richard is currently seconded to the Spanish Council for Scientific Research (CSIC), Barcelona, Spain.

Workshop on Problem Solving Methods IJCAI-97, Nagoya, Japan, 23 August 1997

This report gives the abstracts from each of the paers presented at the workshop, and then a final Analysis section considers the importance of what has been presented for HPKB. The full proceedings of the workshop are available here.

Paper abstracts

Problem Solving methods in Cyberspace: V. Richard Benjamins

The World Wide Web opens the way to scaling up reuse of software components enormously. This paper sketches an approach to remotely configuring a problem solver using reusable libraries of problem solving methods that reside in cyberspace. Two research issues are involved: (1) how to configure PSMs from different libraries for a specific application, and (2) how to deal wiuth the distributiveness introduced by situating reuse on the digital higway. Full paper

Problems in indexing Problem Solving Methods: Joost Breuker

Organizing PSM into a some library for reuse reveals two related problems that have thus far received little explicit attention:

- What is the scope of reuse of a PSM?
- How distinctive/similar are PSM?

In considering the first question, the working hypothesis in constructing such libraries, and in particular the CommonKADS library, has been that PSM are task specific, and that therefore the major indexing is by task types, and in particular by some taxonomy of tasks; secondary indexing is by other assumptions that are inherent to the PSM, in particular related to domain knowledge features. In [Breuker, 1994b] it was argued that such taxonomy is impossible to construct and that a suite of dependent generic types of problems is a better representation to characterize what a PSM should yield. In this paper some revisions of this suite are proposed. Although this enables a clearer functional indexing of PSM (i.e. by their competence), it is of only tertiary importance where it concerns the indexing of the real, operational PSM that heuristically exploit experience-based domain knowledge. The primary indexing proposed here is by classifying the

PSM by the way they generate solutions: by classication or by construction, conforming views held by [Clancey, 1985; de Velde, 1988; Puppe, 1993] Full paper

Assumption Hunting as Developing Method for Problem-Solving Methods: Dieter Fensel and Arno Schonegge

Problem-solving methods (PSMs) for knowledge-based systems need to make assumptions to provide effective and efficient problem solving: assumptions about the scope of the problem they should solve and assumptions about the domain knowledge they can use as a resource for their reasoning process. If these assumptions are made explicit they can improve the reusability of PSMs by guiding the refinement process of problem-solving methods for a given application and by defining goals for the acquisition process of domain knowledge. However, making the underlying assumptions explicit is not an easy task. The goal of our paper is to contribute to solve this problem. The main idea is to construct mathematical proofs and analysis of their failure as a systematic means for formulating assumptions. Tool support is provided by adapting the Karlsruhe Interactive Verifier (KIV) for our purpose. KIV is an interactive theorem prover that returns with open goals if a proof could not be completed. These open goals can be used to derive the assumptions we are looking for. Full paper

An Ontology-based Broker: Making problem-solving Method Reuse Work: Dieter Fensel

We present the architecture of an intelligent broker for enabling the use of problem-solving methods via the World Wide Web (WWW). The core component of such a broker is realised by an ontologist and an adapter. The *ontologist* helps to mediate between domain-specific requirements and knowledge at the one hand and method-specific terms describing the competence and requirements of the reasoning components via generic problem descriptions at the other hand. *Adapters* are necessary to provide domain knowledge and case data to problem-solving methods and to rephrase the output of problem-solving methods into domain-specific terms. They ensure the mapping during runtime. Therefore, the ontologist mediates the *selection* and *adaptation* process of PSMs whereas the adapter mediates the *execution* of them.

Domain-oriented library of scheduling methods: Design principle and real-life application: Masahiro Hori and Taketoshi Yoshida

This paper presents a library of scheduling methods dedicated to production scheduling, and reports our experience in applying the library to the development of a real-life scheduling system. The library consists of three loosely coupled subsystems: a manufacturing domain model, scheduling methods, and a graphical user interface. The domain model provides fundamental concepts to be employed by the other two subsystems, and facilitates integration with other information systems. Applications developed with the library can be adapted to their evolution by modifying configurations of the subsystems and elaborating concepts in each subsystem. The reusability of the library is investigated in terms of the ratio of code reuse in the real case. Full paper

Ripple-Down Rationality: A Framework for Maintaining PSMs: Tim Menzies and Ashesh Mahidadia

Knowledge-level modeling can be characterized as *theory subset extraction* where the extracted subset is consistent and relevant to some problem. Theory subset extraction is a synonym for Newell's principle of rationality, Clancey's model construction operators, and Breuker's components of expert solutions. In an abductive framework, a PSM is the extraction controller and is represented by a suite of BEST inference assessment operators. Each BEST operator is a single-classification expert system which accepts or culls a possible inference. PSMs can therefore be maintained by ripple-down-rules, a technique for maintaining single-classification expert systems. Full paper

Problem Solving for Redesign: Anita Pos, Hans Akkermans and Remco Straatman

A knowledge-level analysis of complex tasks like diagnosis and design can give us a better understanding of these tasks in terms of the goals they aim to achieve and the different ways to achieve these goials. In this paper we present a knowledge-level analysis of *redesign*. Redesign is viewed as a family of methods based on some common principles, and a number of dimensions along which redesign problem solving methods can vary are distinguished. By examining the problem-solving behavior of a number of existing redesign systems and approaches, we came up with a collection of problem-solving methods for redesign and developed a task-method structure for redesign. In constructing a system for redesign a large number of knowledge-related choices and decisions are made. In order to describe all relevant choices in redesign problem solving, we have to extend the current notion of possible relations between tasks and methods in a PSM architecture. The realization of a task by a PSM, and the decomposition of a PSM into subtasks are the most common relations in a PSM architecture. However, we suggest to extend these relations with the notions of *task refinement* and *method refinement*. These notions represent intermediate decisions in a task-method structure, in which the competence of a task or method is refined without immediately paying attention to the operationalization in terms of subtasks. Explicit representation of this kid of imntermediate decisions helps to make and represent decisions in a more piecemeal fashion. Full paper

PSM Modeling and Self-Adaption in Robotics Applications: Eleni Stroulia and Ashok K. Goel

In this paper, we focus on the use of PSM speciifcations in the task of system adaptation, when there is a discrepancy between the behavior delivered by the system and the behavior desired of it. This task arises when a system is re-deployed in an environment different to the one in which it was originally designed, or when the requirements on its behavior evolve during a ``long lifetime". Autonomous robots are but one example of such systems. They usually live in highly dynamic environments and need to be able to adapt to the evolving requirements that these environments impose on their behavior. We describe (i) a language for specifying the PSMs employed by a system and (ii) a porocess, which uses such a specification, to monitor and flexibly control a system's behavior at run-time, and to adapt it when it fails. AUTOGNOSTIC, a shell that implements this modeling language and process, was integrated with two different robots. From these integrations, we discuss two experiments. Full paper

Discussion and Analysis for HPKB

Two of the papers described above, by Richard Benjamins and Dieter Fensel, discuss the subject of using problem-solving methods which are distributed across the Web; this fits well with one of the stated aims of the Problem Solving Methods working group. Both discuss the reuirements for a "software broker" which will automatically search for and find relevant PSMs; Benjamins' chosen approach for the broker is "to add specific tags to HTML documents that give the broker access to the relevant information ... we will need tags to express the *competence* and *domain requirements* of PSMs and that a PSM *realises* a task.", while Fensel separates the broker into four parts:

- a user interface for the client;
- an automated intrface for the library provider (the server);
- an *ontologist* which is expected to use an ontology of problem types which can be matched against the PSMs' ontology/ies;
- an *adapter* which is intended to link different ontologies: domain ontology to the problem ontology, and problem ontology to the PSM ontology.

Benjamins' suggested tags provide useful input for the HPKB work on defining capabilities of problem solving methods, while Fensel's work goes deeper in suggesting the need for a problem-type ontology to mediate between a PSM ontology and a domain ontology; he states that "most other approaches deal with this [2-stage] mapping only as a side-aspect and focus on connecting domain and PSM directly". The possible need for a mediating ontology of problems between domain ontologies and PSMs should be considered carefully by the HPKB problem solving methods. Joost Breuker's paper is also important to the capability description task, particularly so given that Breuker was the architect of the original KADS-1 hierarchy of task types which has been widely accepted in Europe as a way of classifying problem solving methods. Breuker suggests a simpler top level classification of problem types, which are considered to be sequential in the real world (modelling -> design -> assignment -> prediction -> monitoring -> diagnosis); he also highlights the fact that problem solving methods need to be classified according to their *solution type* as well as their task type. The three solution types which Breuker distinguishes are the **case model** which explains the data, the **argument structure** which justifies the conclusion, and the **conclusion** itself. For example, a problem solving method to diagnose a single faulty component (conclusion) will differ from a PSM to diagnose a faulty state (case model), and neither may be able to explain the diagnostic process (argument structure) well.

Of the other papers:

- Hori & Yoshida and Pos *et al* propose new or revised (conceptual level) PSMs for design and scheduling tasks. The growing international library of conceptual level PSMs is still weak on these task types, and new additions to the library are proposed fairly frequently (see Kingston I*et al.* 1996, for example).
- Fensel and Schonegge hypothesise that all PSMs require assumptions to help them solve intractable problems, and they apply a theorem proving program to formal specifications of PSMs in dynamic logic in order to discern these assumptions. For HPKB, it is worth considering what the assumptions associated with a PSM might be, since these are likely to affect the applicability of a PSM to a particular problem.
- Menzies and Mahidadia present a novel application of their "ripple down rules" approach - to maintain PSMs. It's unclear how useful this approach is in practice.
- Stroulia and Goel focus on automatic adaptation of a PSM to solve a different but related problem, using robotics as a test domain.

B.1 Workshop on Problem Solving Methods at IJCAI-97

Publication details: Published on the Web, late 1997. http://www.aiai.ed.ac.uk /project/hpkb/ijcaipsm.html. This document constitutes deliverable N-1.

John Kingston

Purpose: To update HPKB participants on the latest research in the field of problem solving methods.

Abstract: This newsletter gives the abstracts of each of the papers from the workshop. A final analysis considers the importance of what has been presented for HPKB participants.

B.2 Report on EKAW-97: The European Knowledge Acquisition, Modelling & Management Workshop

Publication details: Published on the Web, early 1998. http://www.aiai.ed.ac.uk /project/hpkb/ekaw.html. This document constitutes deliverable N-2.

John Kingston

Purpose: To update HPKB participants on the latest research in the fields of knowledge management, ontologies, problem solving methods and knowledge acquisition.

Abstract: This newsletter gives the titles and authors of all papers, and reviews papers of particular interest to HPKB participants in detail.

This is a report on those aspects of EKAW-97 (the European Knowledge Acquisition, Modelling & Management Workshop) which may be of interest to HPKB participants.

List of published papers

Long papers; these were presented in seven sessions over four days. The themes of the sessions were:

1. Knowledge Management
2. Ontologies
3. Design and Planning
4. Problem-Solving Methods I
5. Problem-Solving Methods II
6. Ripple-down Rules and Verification
7. Knowledge Acquisition from Text

The titles of full papers were as follows:

Session I, Knowledge Management

Information Tuning with KARAT: Capitalizing on Existing Documents
Bidjan Tschaitschian, Andreas Abecker, Franz Schmalhofer
Building up and Making Use of Corporate Knowledge Repositories
Gian Piero Zarri, Saliha Azzam
An Enterprise Reference Scheme for Integrating Model Based Knowledge Engineering and Enterprise
Modelling
Stefan Decker, Manfred Daniel, Michael Erdmann, Rudi Studer

Invited talk
Collaborative Knowledge Management: Building Organizational Memory through On-line Knowledge Work
Tom Gruber

Sisyphus-III
Introduction by Nigel Shadbolt and presentations by Michael Erdmann, Machiel Jansen, Paul Compton
and Jose Luis Sierra.

Sisyphus-IV
Rudi Studer

Session II, Ontologies

An Ontology Approach to Product Disassembly
Pim Borst, Hans Akkermans
Designing Operators for Constructing Domain Knowledge Ontologies
Rodrigo Martinez-Bejar, V. Richard Benjamins, Fernando Martin-Rubio
Acquisition of Conceptual Structure in Scientific Theories
Dean Jones, Ray C. Paton

Session III, Design and Planning

Problem Solving for Redesign
Anita Pos, Hans Akkermans, Remco Straatman
Knowledge Refinement for a Design System
Robin Boswell, Susan Craw, Ray Rowe
Knowledge Acquisition for the Onboard Planner of an Autonomous Spacecraft
Benjamin Smith, Kanna Rajan, Nicola Muscettola

(KA)2 initiative
Dieter Fensel

Session IV, Problem-Solving Methods I

The Tower-of-Adapter Method for Developing and Reusing Problem-Solving Methods
Dieter Fensel
A Systematic Approach to the Functionality of Problem-Solving Methods
Pascal Beys, Maarten van Someren
Reuse of Problem-Solving Methods and Family Resemblances
Rainer Perkuhn

Session V, Problem-Solving Methods II

The Notion of Role in Conceptual Modeling
Chantal Reynaud, Nathalie Aussenac-Gilles, Pierre Tchounikine, Francky Trichet
Using Ontologies For Defining Tasks, Problem-Solving Methods and Their Mappings
Dieter Fensel, Enrico Motta, Stefan Decker, Zdenek Zdrahal
Specification of Flexible Knowledge-Based Systems
Christine Pierret-Golbreich, Xavier Talon

Invited talk
The DARPA Project on High Performance Knowledge Bases (HPKB)
John Kingston

Session VI, Ripple-Down Rules and Verification

Knowledge Acquisition First, Modelling Later
Debbie Richards, Paul Compton
Acquisition of Search Knowledge
Ghassan Beydoun, Achim Hoffmann
Compositional Verification of Knowledge-Based Systems: a Case Study for Diagnostic Reasoning
Frank Cornelissen, Catholijn Jonker, Jan Treur

Session VII, Knowledge Acquisition from Text

An Empirical Evaluation of a System for Text Knowledge Acquisition
Udo Hahn, Klemens Schnattinger
Syntactic Parsing as a Knowledge Acquisition Problem
Sean Wallis, Gerry Nelson

There were also some short papers, presented in poster sessions:

21.Modelling Competitive Co-operation of Agents in a Compositional Multi-Agent Framework - *Frances Brazier, Pascal van Eck, Jan Treur*
22.An Instrument for Purpose Driven Comparison of Modelling Frameworks - *Frances Brazier, Niek Wijngaards*
23.Knowledge Discovery in Rule Bases - *Timo Breidenstein, Isabelle Bournaud, Francis Wolinski*
24.Supporting Probability Elicitation by Sensitivity Analysis - *Veerle Coupe, Linda van der Gaag*
25.Towards More Collaboration Between Machine Learning Systems and their Users - *Jean-Marc Gabriel*
26.COATIS, an NLP System to Locate Expressions of Actions Connected by Causality Links - *Daniela Garcia*
27.Using Knowledge Acquisition to Build Spoken Language Systems - *Stefan Kaspar, Achim Hoffmann*
28.An Assistant for a Design Project: Application to the Design of a Mixed Hardware/Software Architecture - *Laurent Maillet-Contoz, Jean Sallantin*
29.REVINOS : An Interactive Revision Tool Based on the Concept of Situation - *Luc Poittevin*
30.KIDS for KADS - *Remco Straatman*
31.Exploiting Inductive Bias Shift in Knowledge Acquisition from Ill-Structured Domains - *Luis Talavera and Ulises Cortes*

Papers of interest to HPKB

The whole conference was interesting and in formative, with a number of issues raised which may affect or contribute to the work of various HPKB participants. However, the sessions which contained most of relevance to HPKB were the sessions on Ontologies and Problem-Solving Methods.

Ontology papers

An Ontology Approach to Product Disassembly: *P. Borst and H. Akkermans, University of Twente, Netherlands*
Borst and Akkermans described work done by Pim Borst in his Ph.D. (http://www.cs.utwente.nl/~borst/ontology). They have developed an approach to viewing the world in which different perspectives extend, specialize, or a simply provide a view on each other or on real-world objects. For example, systems theory can be *specialized* into processes or into components, booth of which provide *views on* Physical Systems; engineering maths provides an alternative *view onto* Physical Systems; and mereology *extends* topology, which *extends* systems theory. They test out this approach by modelling an electric car as a system (in terms of general systems theory) and product disassembly analysis (not described at the conference, but described in the thesis).

From this, they distinguish 3 types of ontology: abstract ontologies (mereology, topology, systems theory), domain ontologies (cosnsiting of different viewpoints on a physical system), and viewpoint ontologies (components, physical processes, maths).

They use their approach to do cost benefit analysis on different approaches to product disassembly & recycling.

Rodrigo Martinez-Bejar: **Designing Operators for constructing domain knowledge ontologies**

This talk focused on applying operators to concepts elicited from text to check on consistency, inheritance, etc. It's a potentially useful approach if it could be automated; at present, both the extraction of concepts from text and the checking are done by hand.

Dean Jones and Ray Paton: **Acquisition of Conceptual Structure**

The basis of this is that many scientific theories are based on an underlying metaphor, and the metaphor is based on an isomorphic mapping between conceptual models

They describe "image-schemata" which supposedly enable use of prior knowledge in knowledge acquisition without being domain specific

Tom Gruber: **Collaborative Knowledge Management**

Although Tom's talk was included in the Knowledge Management section, the content is relevant to those interested in ontologies. The reason is that Tom's suggested approach to collaborative KM is to create a "group memory", which is a shared repository which can be viewed by everyone on an organisation (through an intranet). Everything in the group memory is labelled by whoever puts it in there; others may add new labels, or subcategories to existing labels. The collection of labels can be viewed by all, and eventually becomes an ontology which (by definition) is not imposed top-down. Gruber's rationale for this is that ontologies are social contracts: agreements about a domain of discourse created for specific purpose, and the main purpose of global taxonomies is to help *find* things. Gruber demonstrated his approach using a software tool called Intraspect, in which the purpose of categories is to capture the *context of use* of information. For example, information about a competitor's products would probably be stored under a subcategory of **Marketing:competition**, rather than in a mereonomic or taxonomic hierarchy.

Not surprisingly, this talk provoked discussion about the advantages and disadvantages of an "organic" ontology versus a designed ontology. Gruber argued persuasively that the chief advantage of an organic ontology is that, because people have created it to meet their needs, it will be used. His philosophy is that it's unrealistic to say "if you build it, they will come"; instead, he argues that "if they use it, it will build itself". If people can be relied upon to build an ontology which has reasonable internal consistency, Gruber's approach seems sound. However, a quotation which was presented in a later paper demonstrates an ontology notably lacking in such consistency:

"Animals are divided into (a) those that belong to the emperor, (b) embalmed ones, (c) those that are trained, (d) suckling pigs, (e) mermaids, (f) fabulous ones, (g) stray dogs, (h) those that are included in this classification, (i) those that tremble as if they were mad, (j) innumerable ones, (k) those drawn with a very fine camel's hair, (l) others, (m) those that have just broken a flower vase, (n) those that resemble flies from a distance. (from Borges, J.L., *Other Inquisitions*, New York, 1966, p. 108, quoting from the Chinese "Celestial Emporium of Benevolent Knowledge")

In other words, it is possible that an ontology designed as needs arise is in danger of creating many overlapping categories which may not be very helpful when used in combination.

Dieter Fensel: **KA^2 initiative**

Dieter Fensel, one of the EKAW organising committee, presented an initiative to develop and use an ontology for knowledge sharing within the knowledge acquisition community. The principle is that KA researchers will mark up their own Web pages with ontological tags, and a search engine at the Ontolingua web site will then allow retrieval of detailed information about the activitries of other knowledge acquisition researchers. An early version of the system is now running - see http://www.aifb.uni-karlsruhe.de/WBS/broker/KA2.html.

Problem Solving Methods

It's worth stating at the beginning of this section that in European AI conferences, there is a default assumption that problem solving methods are not implemented (i.e. are represented in diagrams or text), and are (usually) domain independent. These methods are intended to provide guidance for knowledge engineers, and much research is directed into making the methods more expressive, more generic, or more understandable. The HPKB PSM group makes the default assumption that PSMs should be implemented as single or multiple pieces of code, and is directing its efforts into devising a language which allows interoperability of PSMs. AIAI's view is that for maximum usefulness, the two approaches should be used together.

Dieter Fensel: **The Tower-of-Adapter problem**

Fensel proposes refinement of PSMs using adaptation - while the PSM strategy remains the same, the PSM is made more specific by increasing ontological commitment. The talk includes an example where a PSM for local search (specified in a semi-formal language) is specified to hill climbing, and then further specified to set minimisation (a problem-specific adaptation of hill-climbing). Fensel then discusses the task of abductive diagnosis, and proposes four refinement operators.

Fensel's overall aim is to use this approach to organise a library of problem-solving methods. While it isn't a complete system, his approach holds promise.

A Systematic Approach to the Functionality of Problem-Solving Methods:
Pascal Beys, Maarten van Someren

Maarten van Someren presented a formalisation of PSM goals based on set algebra. The aim of this work is to systematically characterise other PSMs for the purpooses of re-use of PSMs. However, they acknowledge that set algebra "is not the easiest language to describe functionalities", and their work doesn't seem to have progressed as far as Fensel's.

Rainer Perkuhn, AIFB: **Re-use of PSMs and family resemblances**

Perkuhn describes an approach to configuring PSMs for a particular problem based on the "family resemblances" ideas of Rosch. The basic idea appears sound, but Perkuhn hasn't yet developed it in much detail. An alternative source of information on configuring PSMs is a recent Ph.D, thesis from the University of Amsterdam by Annette ten Teije (http://www.cs.vu.nl/~annette/postscript/thesis.ps.Z) - chapter 7 is entitled "Construction of Methods as Parametric Design".

Reynaud, Univ. Paris-Sud et al: **The notion of Role in Conceptual modelling**

In CommonKADS, the most influential knowledge modelling methodology in Europe, the major components of problem-solving methods are inference steps which must be taken in problem solving and knowledge roles - i.e. roles played by domain knowledge in the problem solving process. Nathalie Aussenac-Gilles discussed the use of knowledge roles, noting that they are not just labels but have semantic and syntactic properties (though an agreed definition of these properties is lacking), and that their primary use is in top-down knowledge modelling, where they are used to acquire additional knowledge (by asking "what domain knowledge should play this role?"), to test and check the model (by making sure that each inference step has appropriate inputs & outputs), and to describe generic components and make the process of reuse easier.

Fensel et al: **Using Ontologies for Defining Tasks, PSMs, and their Mappings**

This paper may be of particular interest to the HPKB PSM working group, as it presents a new perspective on a PSM specification as being a method ontology, and shows how this approach could work in practice. The heart of the approach is to view PSMs as search strategies (cf. Fensel's "Tower-of-adapter" paper mentioned above), characterised in terms of the assumptions they make about the viability and properties of domain knowledge. A case study characterises the Propose & Revise PSM in these terms; to summarise the result of the case study, the description of competence of

B.3 Workshop on Ontologies at ECAI-98

Publication details: Published on the Web, http://www.aiai.ed.ac.uk/ project/hpkb/ecai.html. This document constitutes deliverable N-3.

Stuart Aitken

Purpose: To update HPKB participants on the latest research in the fields of ontologies and problem solving methods.

Abstract: This newsletter gives the titles and authors of all papers, and reviews papers of particular interest to HPKB participants in detail.

Report on the ECAI 98 Workshop on Applications of Ontologies and Problem-Solving Methods, 24–25 August 1998, Brighton, UK.

This report summarises a number of papers presented at the Workshop on Applications of Ontologies and Problem-Solving Methods held at the European Conference on Artificial Intelligence (ECAI) 1998. The workshop addressed the application of ontologies, the reuse of problem-solving methods (PSMs), and the integration of ontologies and PSMs.

Ontologies

Where are the killer apps ? Uschold, M. A scheme for analysing ontologies along a number of dimensions is presented. The aim is to compare ontologies and their application, and to identify any inherent technical barriers. The dimensions include *purpose*, *representation language*, *meaning and formality*, *subject matter*, *scale*, and *development* i.e. whether the ontology has been fielded. It is noted that in AI many ontologies are for research purposes, are relatively small-scale, and formally specified, while in linguistics and in information retrieval applications the scale is larger and the emphasis on formality and meaning reduced. A characterisation of existing ontologies in these terms would assist potential users of ontologies to gain an understanding of the field.

(ONTO)^{2}Agent: An ontology-base WWW broker to select ontologies Vega, J.C.A., Gómez-Pérez, A., Tello, A.L., and Pinto, S.A.N.P. Three problems in the reuse of ontologies are identified: the lack of standard features characterising ontologies from the users' point of view, that web sites use different organisational approaches and present different information, and, thirdly, that the search for appropriate ontologies is hard, time-consuming and usually fruitless. The paper presents a set of descriptive features and shows how they are used in a WWW retrieval broker. The descriptive features constitute a *reference ontology*, and several feature-types are similar to the dimensions proposed by Uschold.

Automatically acquiring requirements of business information systems by reusing business onto Jin, Z., Bell, D., and Leahy, D. An automated approach to eliciting the requirements of a business information system is presented. A business ontology and a domain ontology are used to acquire the customer's requirements - without the use of software engineering terminology.

Using multiple ontologies in a framework for natural language generation Frölich, M. and van de Reit, R. A system for natural language generation (NLG) from structured knowledge sources is described. Knowledge is represented at three different levels: content determination, sentence planning, and sentence generation. The upper levels make use of a domain-dependent ontology of concepts, while the lower level uses a domain-independent linguistic ontology. It is argued that this separation of domain-specific and generic knowledge is beneficial in NLG applications.

Problem-Solving Methods

Ontology of tasks and methods Chandrasekaran, B., Josephson, J.R., and Benjamins, V.R. It is argued that generic problems solvers have five identifiable elements: a goal, a description of the problem instance, the problem-solving state, problem-solving knowledge, and domain factual knowledge. Domain factual knowledge is viewed as the traditional focus of ontology, and the need for shared concepts of problem-solving tasks and methods is pointed out. An ontology can be associated with each element of a problem solver. For example, the problem-solving goal has 'attitude terms' which can be further specialised into terms such as *explain*, *identify*, and *classify*. Problem-solving knowledge is represented as mappings from conditions on goals, states and problem instance data, to changes in problem state. Each of these types of knowledge is represented by terms from the appropriate ontology. Parallels are drawn with Generic Tasks, CommonKADS, and Prótegé.

-IBROW3- An intelligent brokering service for knowledge-component reuse on the World-Wide V Benjamins, V.R., Decker, S., Fensel, D., Motta, E., Plaza, E., Schreiber, G., Studer, R., and Wielinga, B. The IBROW3 project aims to develop an intelligent brokering service for the construction of problem-solving methods. It is intended to provide access to a library of PSMs and to support the tasks of PSM selection, configuration, and adaptation. Configuration involves the combination of PSMs, while adaptation involves relating the configured problem solver to the domain ontology.

An ontology for building a conceptual problem solving model Ikeda, M., Seta, K., Kakusho, O., and Mizoguchi, R. This paper describes the use of a task ontology in a programming environment for problem-solver design. The advantages claimed for this approach are that users can describe their problem solving in high-level terms, task execution can be observed, and there is continuity from the user's description to the computational level. The task ontology provides the vocabulary for representing the problem-solving process. It includes a number of more specific ontologies including generic verbs, for example, *assign* and *select*, and generic nouns. The verb + noun combination constitutes a generic process. The authors view the task to be performed as an important issue in ontology design and apply a task viewpoint when specifying ontologies to represent domain knowledge. This approach differs from that of Chandrasekaran *et al.* and from Benjamins *et al.* who take the conventional task-independent view when considering the organisation of domain knowledge.

C Knowledge models and knowledge acquisition results from Year 1

Publication details: The models contained in this document were made available to the HPKB participants in early 1998. These documents constitute deliverable K-1.

John Kingston

Purpose: To support HPKB participants (in both integration teams) in development of solutions to challenge problems.

Abstract: This deliverable contains three documents:

- A case study of movement analysis, in which experts are presented with a movement analysis task, and are required to ask for all necessary information verbally.
- Process models of movement analysis, showing the movement analysis process as a series of task models, represented in IDEF3.
- Process models of workarounds planning, also represented in IDEF3. It can be seen that in terms of the number of tasks to be carried out, workarounds planning is simpler than movement analysis.

Case Study: Movement Analysis
SMEs: Lee Sweetapple & Don Alexander
06 January 98

This document provides a transcript of a "verbal case study", in which the knowledge engineer (John) constructs a movement analysis scenario, and then asks the SMEs to analyse the situation by asking verbally for any required information.

The hypothetical scenario was as follows: in a large country where enemy forces occupy the southern half, a new "special" weapon had just been released from a factory, and two groups of vehicles were heading for the front line in order to deploy the weapon. One group consisted of a convoy of trucks moving northwards on a road (which runs through a north-south valley in a hilly area), the other consisted of several jeeps or other all-terrain vehicles which were travelling across country in order to take the most direct route to the deployment point, which is to the north-west of the factory. It can be presumed that there is a good reason (possibly dispersal of forces) for dividing the transportation effort between small individual vehicles and one large convoy.

For the information of readers, the hypothetical scenario was based on the "Beaujolais run", when vehicles hurry to carry the newly released Beaujolais champagne from the south of France to the English Channel ports. The event is turned into a "race" by tasking vehicles to complete the journey by the shortest possible route (within 24 hours); therefore, while several commercial vehicles carry the champagne northwards on the main roads, a number of smaller off-road vehicles are travelling on minor roads, farm tracks, and so on.

Case Study

Speakers:
John: Knowledge Engineer
Lee, Don: Subject Matter Experts
Jon, Dave: Other HPKB technology developers or integrators

John: The situation is that there is a … you are in the military force, you are within a fairly large country, you occupy the northern half of the country. You observe some unusual vehicle movements within a short period of time in the south of the country heading vaguely northwards.

Lee: Who observed it?

John: MTI did. … What I want you to do is to keep asking questions till you are satisfied

Lee: What was the size of the enemy formation? How many 'hits'?

John: Well, there is one apparent convoy.

Lee: How many blips in the convoy?

John: About 20 or 30. There are also a number of smaller blips which are moving faster in different directions.

Lee Were they moving at a steady speed for the entire time?

John: The convoy is; the smaller blips aren't.

Lee: All right and where were they located? Were they located leaving the city? Were they going towards a large city or leaving a city?

John: They are all leaving a specific point which doesn't appear to be in a city.

Lee: Do the small blips that are going to a specific point do they disappear there - does the appearance of the small blips stop at the same point that the large blips left from?

John: No they all started there and they are all heading north. The small blips are still visible on the screen there. They are heading in roughly the same direction as the larger blips.

Lee: In roughly the same direction?

John: Yes, they are .... the convoy is heading roughly north and the smaller blips roughly north-west.

Lee: And did you say the number of small blips?

John: Difficult to determine because they are on and off the screen but I would say about 15 - say 10 to 20.

Lee: Did you have any previous identification from that location of a particular unit in that area or a garrison activity or any information about the site they had departed from?

John: It seems to be some kind of employment centre, people visit it and depart from it each day. Quite a lot of people. Possibly some kind of industry.

Don: Had you observed this activity previously? Is there any pattern for this activity?

John: No this is a first time occurrence.

Lee: Do you know if the convoy is moving on a road?

John: The convoy does appear to be moving on a road.

Lee: And are you still observing these blips now? Are they still under observation?

John: The smaller ones are still hard to pick up and particularly because they seem to be off road. But the convoy is still easy to find.

Lee: Was there any bombing activity? Is this during a combat situation?

John: Yes

Lee: O.K., was there any bombing activity prior to the movement?

John: No

Don: No hostile activity at all towards ... at that location?

John: No

Lee: You want me to come up with a conjecture as to what is going on there? Or what would be the..........?

John: Unless you want to ask for any more information

Lee: Based on MTI ... I am interested about the reliability of the sensor. How far was the sensor from the target location? How long did the MTI observe the target?

John: The MTI was about the limit of his range .... but it had observed the target for half an hour.

Lee: And they are still moving as far as you know?

John: They are still moving. They are moving towards the MTI so the track is getting a little bigger.

Lee: And you are limiting me to this one source of information. You are acting like you are up in the JSTARS reporting this or ... ?

John: I could be all sources of information.

Lee: Do we have any SIGINT reports of this?

John: There is some local radio traffic between the convoy. There is very little from the smaller group.

Lee: Is there any radar activity?

John: There is some radar activity from previously known fixed sites.

NOTE: With hindsight, this was not the best answer to give. A better answer would have been that there was noticeable radar activity around the factory site, since important production facilities are usually protected by radar and air defense systems. It turned out that one of Lee's motivations in asking this question was to investigate whether there was anything special about the factory site.

Lee: Is it daylight?

John: No, it is night.

Lee: Night time. Do we have any HUMINT sources along the road?

John: We have one or two. The ones along the road report trucks, traffic ... convoy.

Lee: Are they moving with their lights off or on?

John: Off. We have also had one or two human reports of smaller four wheel drive vehicles taking unusual paths across country. We are not quite sure of what they are doing ... they could be taking farm tracks and one of them has even been seen driving across the top of a dam.

Lee: What you basically described is a battalion moving out of an assembly area with some recce elements out front, if this is a combat situation. Lights off tells me it is probably military ... civilian vehicles don't usually move like that convoy although they do sometimes. The scattered vehicles could be reconnaissance, it could be a bunch of troops hot-dogging in their jeeps depending on the country that you are in.

Don: Radio traffic within the 20 or 30 vehicles would tell me that they're tactical units as opposed to one big convoy that would be moving with probably just one radio ... and the fact that they are still moving. How long have we been observing?

John: Getting on for an hour now.

Lee: We should have an approximate speed then.

John: The trucks are doing close to ... well the convoy is going at close to the maximum speed for trucks.

Don: At night!? Man, that's a well-trained ...

Lee: They must be driving with NODs - night vision goggles ...

Don: Some sort of night observation devices ...

Lee: Even then, that's a pain because you can't see the potholes when you are driving with those.

John: Even stranger the things going across country are going almost as fast and some of them seem to be using lights, on and off.

Don: We definitely have got these guys with some kind of night devices which, based on the country you're fighting, that would be an indication of a fairly special unit. Most countries right now don't have a super abundance of might ... night and thermal devices. But normally real convoys don't have a lot radio

Lee: Motorized infantry units? Infantry mounted on trucks?

Don: You wouldn't think you'd put a lot of those guys with ... well, you might put them with night sights ... 20 or 30 is too big of a convoy for those kind of [soldiers?] ... the guy says there's 20 to 30 trucks, sound like you might have close to a motorised rifle regiment in terms of troops ... you can cram 20 or 30 troops on a truck

Lee: Are they going towards an airfield any place?

John: No but they do seem to be heading .... it's like the ones going across country are taking a more direct route and the convoy is taking the road but going to the same place.

Lee: You see you know what I would do to make this more realistic. I would be looking at a map and as soon as you give me the report I'd be placing the report on the map. And I would be looking - "o.k. they departed from this point .. gee what's there, ... they are headed toward this direction. What's in the direction they are headed?"

John: O.K. They are headed for a point a reasonable distance inside our lines.

NOTE: Again, with hindsight, this was not the best answer, since trucks and jeeps are very unlikely to breach the front line. A better reply would have been "they are heading towards one of our strategic defense facilities on the front line"

Don: They are driving towards our forward line.

Lee: Mounted in trucks?

Don: They are not going to get very far ... if we have combat forces, that would be a formidable stop ... I think Lee's point was ... the one that I would be looking for, especially if I thought that this was infantry is some kind of forward ... some kind of area reviewing point

Lee: Pick-up point

Don: ... [where] they've got helicopters ... I would say that they are going to pick up for some kind of air mobile operation. I am not absolutely certain why they left an industrial ...

Lee: I can't understand if they were going to a PZ why they'd have recce elements ... why they would have the small movers out there.

Don: That may just be ... On the other hand there may not be a pattern to that movement. It could be totally independent of each other. Your focus of concern here would be those 20 to 30 vehicles moving at great speed without lights.

John: Let me stop the exercise there. I will tell you in a moment what I based the whole thing on. But first let me go back over some of your questions. I would like to ask why you asked some of the things you asked. And actually why you didn't ask one or two things. Why did you ask if there had been bombing in the area?

Lee: I wanted to know if they had been targeted and if they were leaving an operational area ... they were evacuating.

John: Why would they evacuate towards your lines though?

Lee: That far from the lines, they might very well move forward.

Dave: You wouldn't know though if they are moving towards your line at that point.

Don: But if you are at maximum range with the MTI, then you know you had pretty much loads of time before you were going to hit our front.

John: You asked about radar activity. Why was that?

Lee: To see if there was ADA. Because if it was some important target like chemical weapons they would probably have air defense out. They might want to have some radar out to let 'em know if aircraft were responding to their convoy.

John: In that case I should have probably answered yes to that one in the hypothetical situation. You asked if it was daylight or night time. Why was that?

Lee: Security of movement. Because if it was a daylight move it probably wouldn't have been anything too important. If it was a night time move and they were moving with their lights on they were probably civilian vehicles. If they were moving with their lights off they were probably military.

John: You asked about the speed they were doing.

Lee: They were moving much too fast for a military convoy.

John: Military convoys do up to about 50Kph?

Lee: That's pushing it. 30Kph is a good rule.

Don: At night, 20Kph is pretty good. You go out in the dark and crank her up and see how fast you can go.

Lee: What happens if you go much faster than that is that you lose the tail end of your convoy. It starts breaking up on you and you can't control it. And with the radios, moving that fast you could explain that they would talking to each other more than usual because of that very reason.

John: You didn't ask anything about the terrain.

Lee: I asked if they were on a road.

Don. The fact that you had them under continuous track and they were on a road - you're in reasonably flat terrain

Lee: Going that fast they are not going to be in mountainous terrain, if you have them going at 50K.

Don: You wouldn't be able to keep them under continuous track on mountainous terrain ... you'd be breaking ... you'd lose them for periods of time.

John: Actually they were in mountainous terrain but they were in north-south ... a valley that runs north-south.

Lee: They were in mountainous terrain going 50K.

John & Don: In a valley

John: And you didn't ask anything about the weather.

Don: I think realistically you would probably have an idea as you looked outside your tent. The MTI operator would know that. He may be back 250 kilometres from where that's going on. He may or may not know what the weather is in that area.

Lee: I wouldn't expect to get that information from an MTI operator.

Don: You would get that from another source.

John: Right. I will finish the exercise ... Jon on you go.

Jon: I have a question for them at some point about what they were doing ... you see in this case you were coming up with speculations without having any intelligent analysis to guide any expectations. Do you think the answers you were coming up with were useful? Or that ... if we were trying to watch for things that were going that people might want to be alerted of. To what extent do we require ...

Lee: There was such a vacuum of information in this that I would not have gone to wake up the Commander Base on this report. If I had more information that movement along this road was important; or if I had information that there was a helicopter pick-up zone likely in the area and the IPB would have shown where any possible pick-up zones would be. Then it would have become extremely important.

Don: Or if the origin of the movement had something there really significant. Like we thought that this was a base for chemical weapons storage, or something else. The fact that left something that was fairly innocuous ... 20 trucks would have not got my pulse racing.

Lee: 30 tanks might have.

John: If we had made that distinction between armour and wheels. But I agree with Lee this is not something that is going to cause...

Lee: A scout platoon could have taken these guys out if they remain mounted with their .50 cals; I mean 20 - 30 trucks is not much of a threat.

John: Part of the point of the exercise is to see what questions you ask and what order you ask them in. Because it shows what you think is the important things you need to know. And then by going back over the questions, why did you ask about bombing, why did you ask about radar. It brings out aspects that might have not have been otherwise brought out ...... any information you need in order to do your job.

Don: One thing that Lee pointed out when he said that how he would have been answering those questions. He would have been looking at a map and plotting that on the map board. And that map board would visually have conveyed to him ...... just by the nature of his experience a lot of information and probably would have generated more questions.

Lee: We look for what we call SALUTES: *size, activity, location, uniform, time* and *equipment* and then the extra S which is the *source* of the information. And that helps to avoid circular reporting and also establishes the validity of the information.

John: Uniform means Military or non-military does it?

Lee: It means the type of military with as much detail as you can get on that.

Don: Did they change that U? I thought it was Unit

Lee: Unit or uniform.

John: The scenario that I was working on was the source that the place they were coming from was actually producing a brand new type of weapon. And these guys were actually racing to get this weapon to a place where it could be used as quickly as possible. And the information I gave you came the fact that I generalised the whole thing from the Beaujolais run in France. Where people compete to get Beaujolais back from France to Britain in 24 hours by the shortest possible route. Hence the story of the small vehicles cutting .... because they actually pre-planned the most direct route they could find, whereas the bigger vehicles had to take the road. So that is where I got the information from. But the fact it is a unusual one because you had to start asking questions ... you have to ask all the questions you need to know

Dave: Was this new weapon mad grape disease?

John: Well, I put a military flavour on ....

Don: But that is something that again in .... there had been some indicator of its existence otherwise truthfully the fact that you see some trucks moving quickly down a road, in and of itself would never indicate that these guys are racing to use a weapon especially when we ask the question the origin of the movement was this a base ... are there any essential dangers that we currently knew. And they are some of the pitfalls that the INTEL analyst can fall into because you are not going to go in and wake the boss up ... hey 20 trucks are going down the road.

John: Yes, it is an unusual situation. [Laughter and general agreement].

Don: And a real difficult military manoeuvre.

Process models of movement analysis

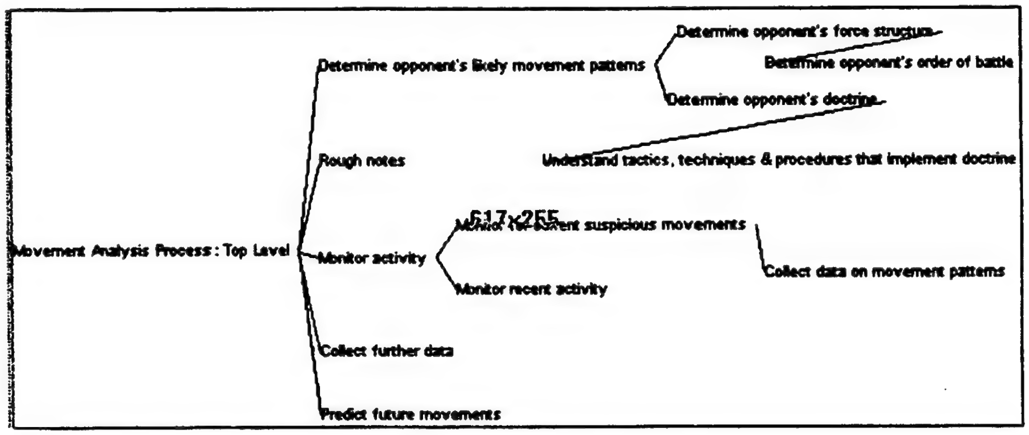
John Kingston, 9 January 1998

This document contains the IDEF3 diagrams which represent the process followed when movement analysis is being performed. These diagrams have been presented to SMEs for constructive criticism, but should be treated as drafts until they have been reviewed more thoroughly.

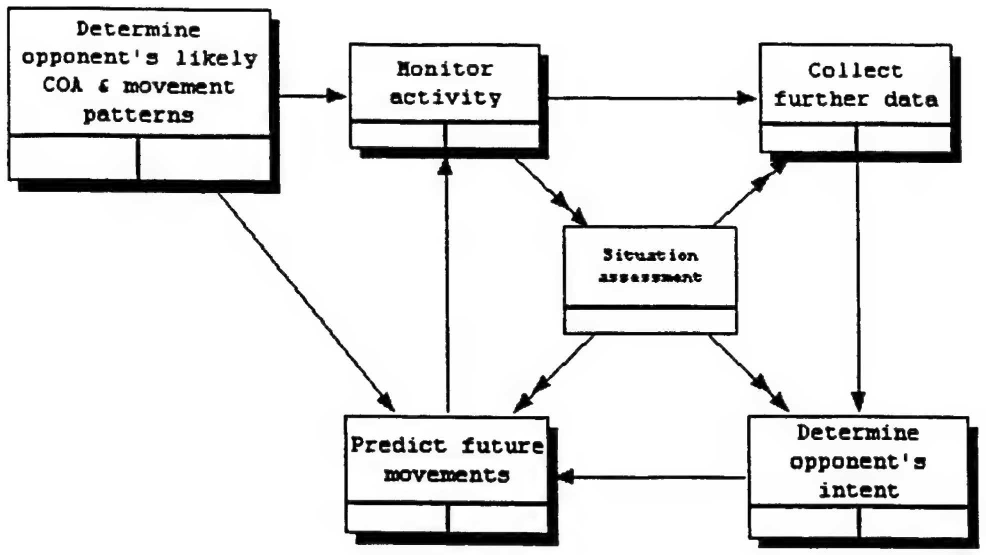
The aim of the models is to represent the process of performing movement analysis, including those activities which HPKB is not planning to support, in a top-down fashion. This allows us to understand the context of the specific activities which we are supporting (those activities connected with monitoring movement activity); to consider the inputs supplied and outputs required for the activities whch HPKB is supporting; and to provide a framework for experts to verify that all relevant aspects of the process are being considered by HPKB, and nothing is being overlooked.

A note on diagram representation: if an activity has a shadow, then it is expected to be decomposed into sub-activities. Some of these sub-activities have been detailed in the diagrams below; other shadowed nodes are considered as placeholders, for which further knowledge could be acquired if required.

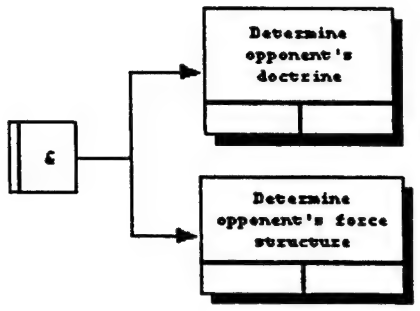
1/ Hierarchy showing how the following diagrams relate to each other


Movement Analysis Process : Top Level
Determine opponent's likely movement patterns
Determine opponent's force structure
Determine opponent's order of battle
Determine opponent's doctrine
Understand tactics, techniques & procedures that implement doctrine
Rough notes
Monitor activity
617x255
Monitor for current suspicious movements
Collect data on movement patterns
Monitor recent activity
Collect further data
Predict future movements

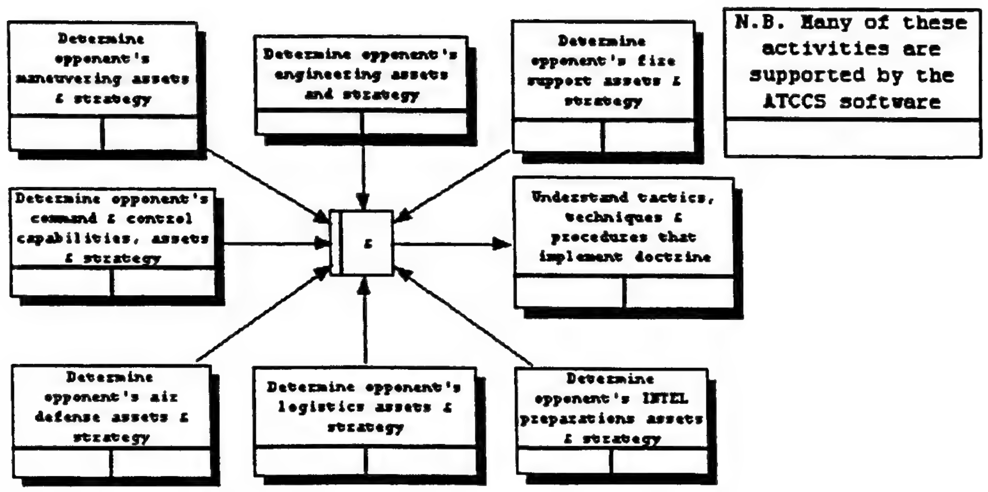
2/ Top level process. Note that "Situation assessment" is a different type of node to the others; rather than representing an activity, it represents a REFERENT to a set of data which is an output of one activity, and forms an input to other activities.



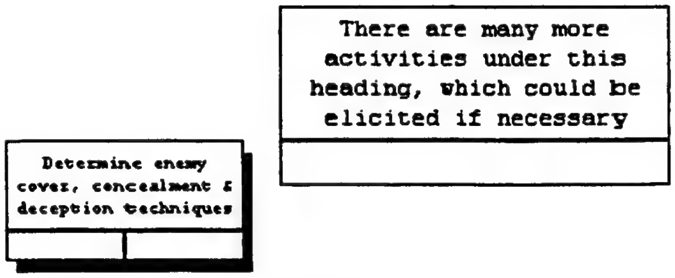
3/ Determine opponent's likely COA and movement patterns. This diagram,and all diagrams following it, decomposes an activity which has appeared in a previous diagram into sub-activities. In this case, there are two sub-activities, each of which has further sub-activities of its own. The & in a box is an IDEF3 asynchronous junction; it represents the fact that both activities must start simultaneously, but need not finish simultaneously.


&
Determine opponent's doctrine
Determine opponent's force structure

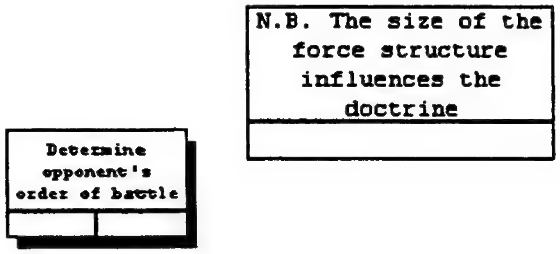
4/ Determine opponent's doctrine. The "referent" box here is simply used as a place-holder for a descriptive note.


Determine opponent's maneuvering assets & strategy
Determine opponent's engineering assets and strategy
Determine opponent's fire support assets & strategy
N.B. Many of these activities are supported by the ATCCS software
Determine opponent's command & control capabilities, assets & strategy
&
Understand tactics, techniques & procedures that implement doctrine
Determine opponent's air defense assets & strategy
Determine opponent's logistics assets & strategy
Determine opponent's INTEL preparations assets & strategy

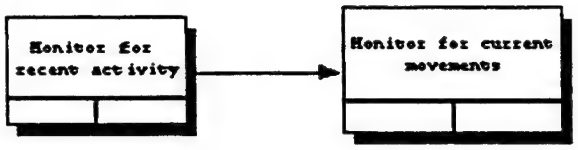
5/ Understand opponent' tactics & techniques


There are many more activities under this heading, which could be elicited if necessary
Determine enemy cover, concealment & deception techniques

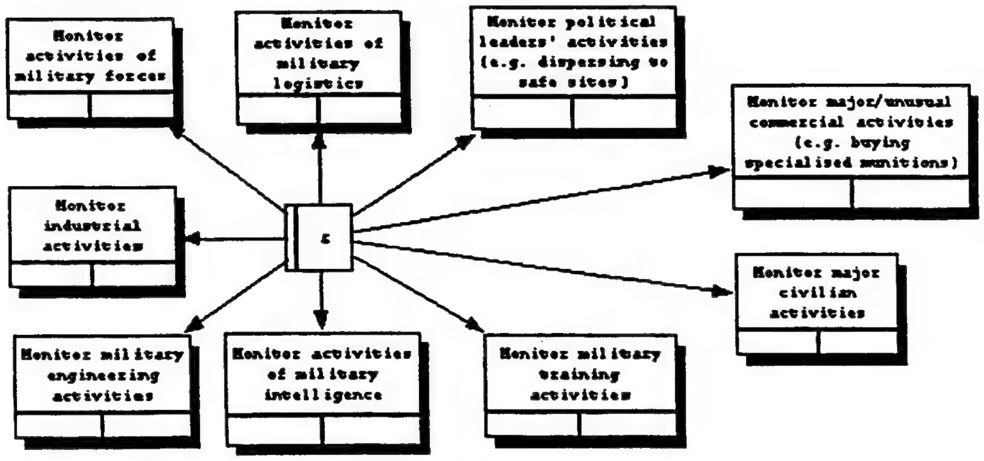
6/ Determine opponent's force structure


N.B. The size of the force structure influences the doctrine
Determine opponent's order of battle

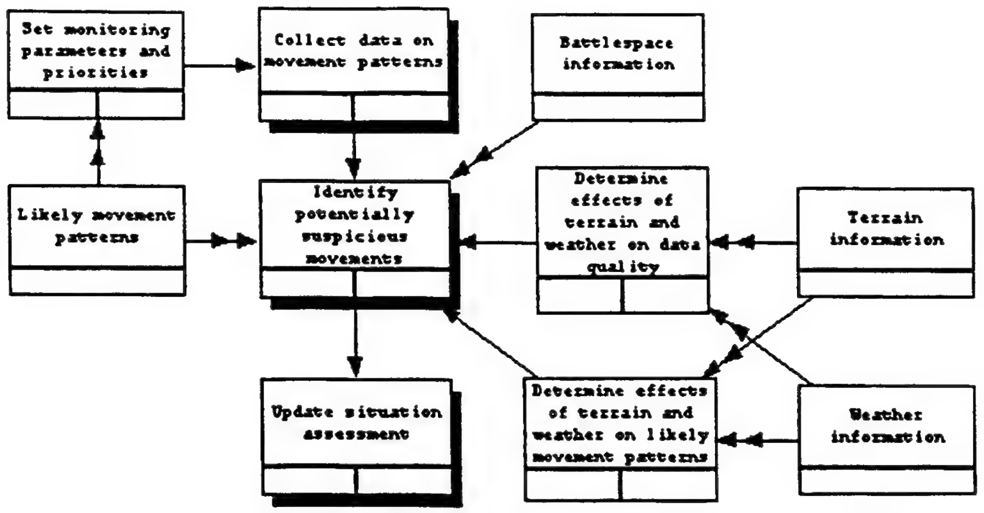
7/ Monitor activity. This diagram, and the ones representing sub-activities of it, represent the activities which are likely to be supported by HPKB.



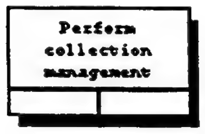
8/ Monitor for recent activity. The purpose of this monitoring is to identify possible preparations for battle - e.g. military maneuvers, increased gasoline refining, civilian evacuations, removal of obstacle to military mobility, etc.


Monitor activities of military forces
Monitor activities of military logistics
Monitor political leaders' activities (e.g. dispersing to safe sites)
Monitor major/unusual commercial activities (e.g. buying specialised munitions)
Monitor industrial activities
&
Monitor major civilian activities
Monitor military engineering activities
Monitor activities of military intelligence
Monitor military training activities

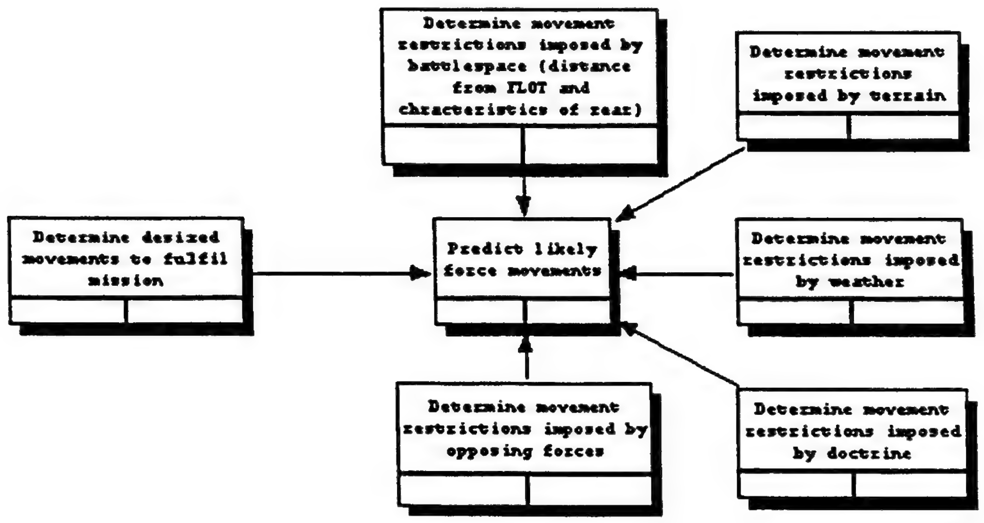
9/ Monitor current activity


Set monitoring parameters and priorities
Collect data on movement patterns
Battlespace information
Likely movement patterns
Identify potentially suspicious movements
Determine effects of terrain and weather on data quality
Terrain information
Update situation assessment
Determine effects of terrain and weather on likely movement patterns
Weather information

10/ Collect further data



11/ Predict future movements.


Determine movement restrictions imposed by battlespace (distance from FLOT and characteristics of rear)
Determine movement restrictions imposed by terrain
Determine desired movements to fulfil mission
Predict likely force movements
Determine movement restrictions imposed by weather
Determine movement restrictions imposed by opposing forces
Determine movement restrictions imposed by doctrine

Other information

Other information which was acquired on movement analysis during this knowledge acquisition session includes:

- A "case study" which will be typed up and distributed
- Information on spotting, and drawing inferences from, a couple of the patterns which the CP spec. asks to be recognised: movements of HETs (heavy equipment transports) and convoys. It is expected that notes on these will be made available.

DRAFT IDEF3 Models of the Workarounds Process

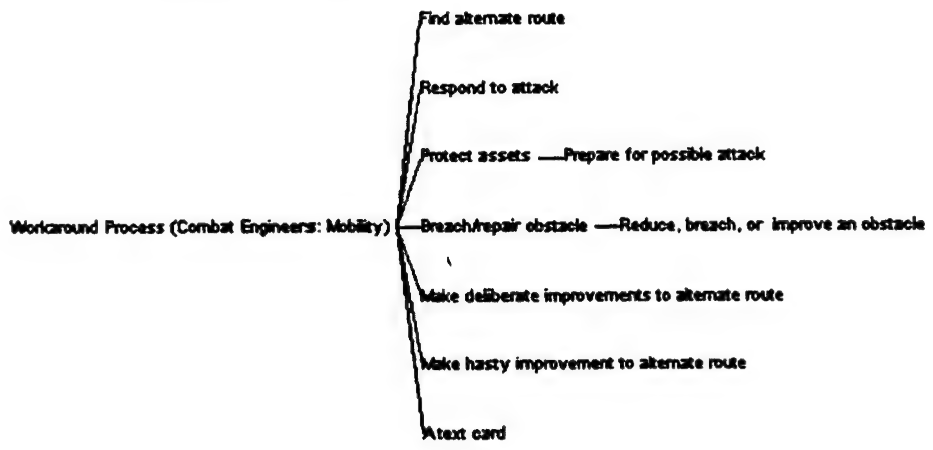
John Kingston, 9 January 1998

This document contains the IDEF3 diagrams which represent the process followed when movement analysis is being performed. These diagrams have been presented to SMEs for constructive criticism, but should be treated as drafts until they have been reviewed more thoroughly.

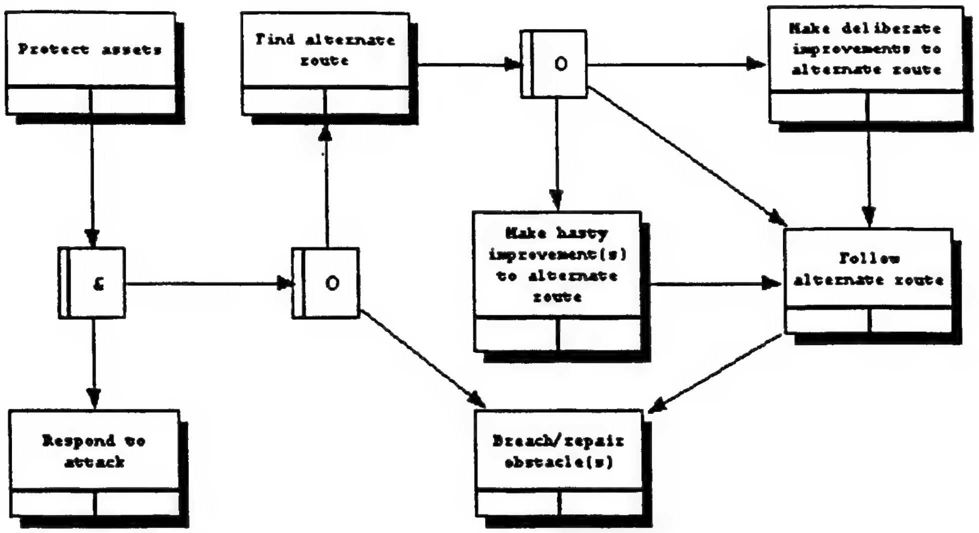
The aim of the models is to represent the process of performing movement analysis, including those activities which HPKB is not planning to support, in a top-down fashion. This allows us to understand the context of the specific activities which we are supporting (those activities connected with monitoring movement activity); to consider the inputs supplied and outputs required for the activities whch HPKB is supporting; and to provide a framework for experts to verify that all relevant aspects of the process are being considered by HPKB, and nothing is being overlooked.

A note on diagram representation: if an activity has a shadow, then it is expected to be decomposed into sub-activities. Some of these sub-activities have been detailed in the diagrams below; other shadowed nodes are considered as placeholders, for which further knowledge could be acquired if required.

Overview of the hierarchy of diagrams

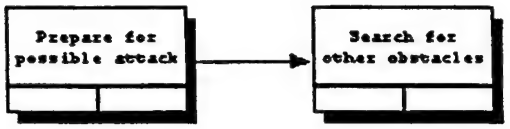

Find alternate route
Respond to attack
Protect assets
Prepare for possible attack
Workaround Process (Combat Engineers: Mobility)
Breach/repair obstacle
Reduce, breach, or improve an obstacle
Make deliberate improvements to alternate route
Make hasty improvement to alternate route
A text card

Workaround Process: Top Level

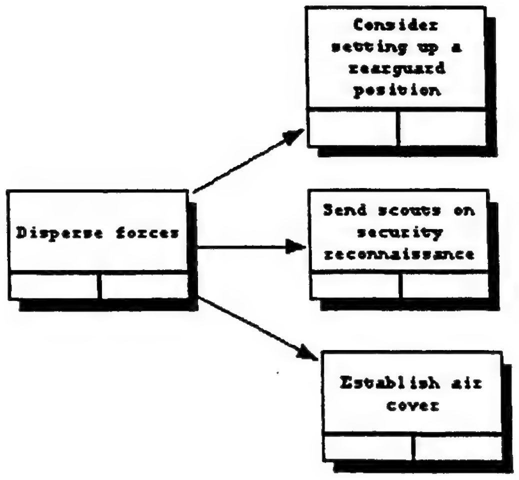

Protect assets
Find alternate route
O
Make deliberate improvements to alternate route
&
O
Make hasty improvement(s) to alternate route
Follow alternate route
Respond to attack
Breach/repair obstacle(s)

Protect assets

N.B. It is likely that the challenge problem (for year 1 at least) will assume that no enemy attack will take place. There may, however, be multiple obstacles, such as mines placed at a fording site close to a destroyed bridge.

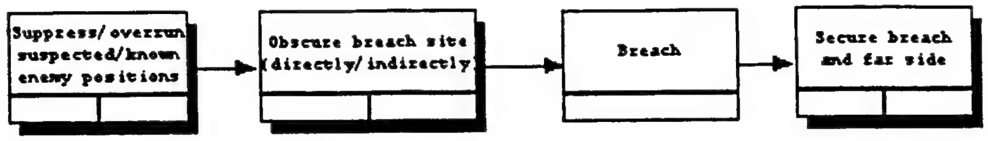


Prepare for attack

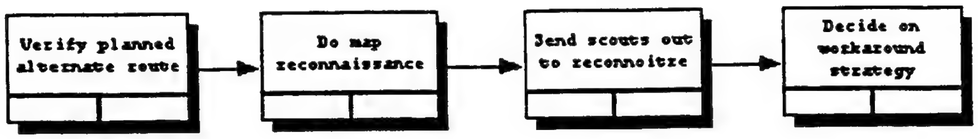

Consider setting up a rearguard position
Disperse forces
Send scouts on security reconnaissance
Establish air cover

Respond to an attack

N.B. "Breach" hee is a REFERENT - it refers to the various activities and sub-activities which are decribed in diagram ?? below.

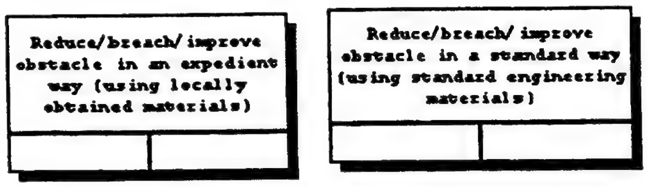

Suppress/ overrun suspected/known enemy positions
Obscure breach site (directly/ indirectly)
Breach
Secure breach and far side

Find alternate routes

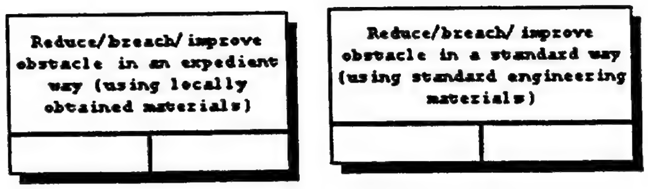

Verify planned alternate route
Do map reconnaissance
Send scouts out to reconnoitre
Decide on workaround strategy

Make hasty improvements

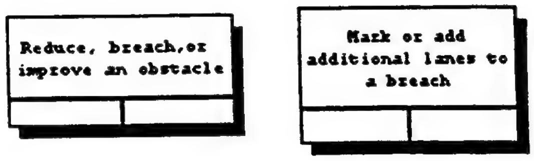
N.B. It was agreed with the expert that breaching/reducing/improving operations could be classified in two ways:
Hasty vs. Deliberate; this is connected with the time taken to construct the improvement. It is usual for hastily constructed breaches to last for a short time before heavy traffic makes them unusable.
Expedient vs Standard: expedient improvements use locally obtained materials (dirt, trees, boulders, even chain link fences) whilst standard improvements use combat engineering equipment (AVLBs, metal matting, girders, etc)


Reduce/breach/ improve obstacle in an expedient way (using locally obtained materials)
Reduce/breach/ improve obstacle in a standard way (using standard engineering materials)

Make deliberate improvements


Reduce/breach/ improve obstacle in an expedient way (using locally obtained materials)
Reduce/breach/ improve obstacle in a standard way (using standard engineering materials)

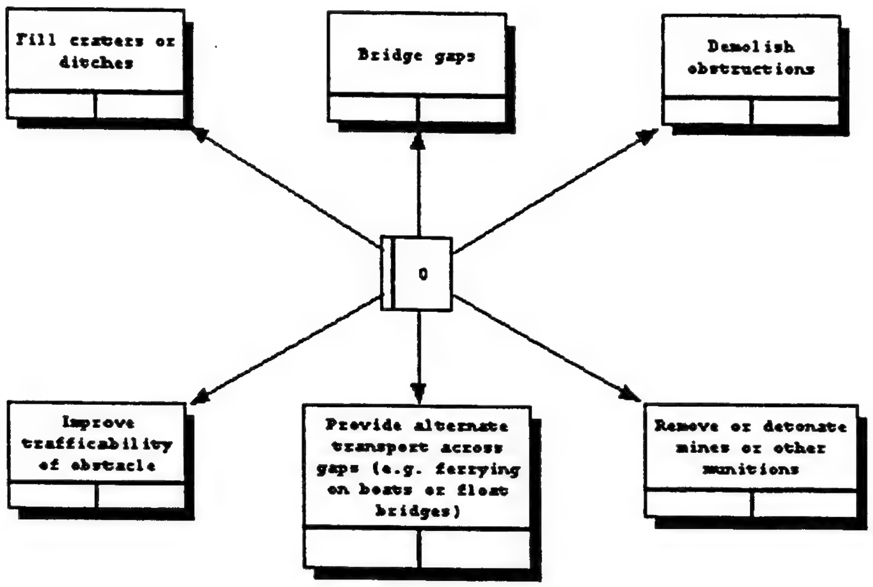
Breach/repair


Reduce, breach,or improve an obstacle
Mark or add additional lanes to a breach

Reduce, breach or improve

This diagram represents all possible categories of workarounds which may be considered. Gaps may be worked around by filling (usually for relatively small "dry" gaps only) or by bridging; obstructions may be demolished; mines (which are considered a special case) can be either removed or detonated; and most types of obstacle may have its trafficability improved or may be overcome using alternative transport.

Often, the first strategy considered when an obstacle in encountered is to bypss it. Bypassing can be considered as a process of route finding, and of breaching, reducing or improving any obstacles which lie across the bypass route - in which case, the options listed below are considered.


Fill craters or ditches
Bridge gaps
Demolish obstructions
0
Improve trafficability of obstacle
Provide alternate transport across gaps (e.g. ferrying on boats or float bridges)
Remove or detonate mines or other munitions

Other information

Other information available on workarounds from this knowledge acquisition session includes a work-through of some of the vignettes from the most recent challenge problem specification, and references to a number of relevant field manuals, especially FM 5-33 and FM 5-34. Some of these manuls can be obtained from the Fort Leaven Wood web site at http://www.wood.army.mil

D Knowledge models and knowledge acquisition results from Year 2

Publication details: The models contained in this document were made available to the HPKB participants in early 1999. This document constitutes deliverable K-2.

John Kingston

Purpose: To support HPKB participants (in both integration teams) in development of solutions to the Year 2 Crisis Management Challenge problem.

Abstract: This deliverable explores the interaction between interests and actions of different actors in a crisis situation. It proposes some scales and tables by which interests and actions can be classified.

>Shoot (specific) <PersonType>
>Blow up (small group of) <PersonType>
>Fatally infect (large subset of) <PersonType>

It would seem that the first is different since it involves violence against one person, while the other choices involve violence against a group of people.

Jill Jermano

>1/
>Economically support <InternationalAgent>
>Militarily support <InternationalAgent>
>Diplomatically support <InternationalAgent>

The first two items involve providing some type of material support; diplomatic support implies some type of non-material assistance, such as a statement, a vote, a concession, etc.

>2/
>Explicitly threaten <InternationalAgent> with <EventSpec>
>Demand <EventSpec> of <InternationalAgent>
>Attempt to initimidate <InternationalAgent> into <EventSpec>

To explicitly threaten and intimidate involves pressuring an <Int'lAgent> with a warning of some type of negative repurcussion if <Int'lAgent> does not acquiesce. A demand is a forceful request with no implied threat.

>3/
>Increase economic aid to <InternationalAgent>
>Provide military aid to <InternationalAgent>
>Initiate managing public health for <InternationalAgent>

This set could have several interpretations, depending on how you define "initiate managing public health." Here's one: the first two involve providing specific types of material assistance whereas the third one refers to creating a program. Or, the first two are foreign assistance actions (implied by the word "aid") whereas the third involves the establishment of a social program within a country by domestic actors.

>4/
>Embargo <ProductType> to <InternationalAgent>
>Curtail flow of <ProductType> to <InternationalAgent>
>Charge fee for use of transport for <ProductType>

Although all three actions are potentially negative actions (unless the need to "curtail flow" arises because of a shortage of or decreased demand for <ProductType>), the first two, which imply an intent to punish, are more negative than the third, which involves a simple transaction cost.

>5/
>Conduct peacekeeping mission
>Conduct search and rescue mission
>Conduct counter-terrorism mission

The first two actions involve carrying out an operation to achieve humanitarian/peaceful goals. Counter-terrorism involves hostile actions designed to defeat terrorists/preclude terrorist acts.

while the other two involve the promotion of a policy.

>14/
>Rights to <ProductType>
>Infrastructure for <ProductType>
>Price of <ProductType>

The second is different since it involve a physical thing, infrastructure items, while the other two involve non-physical items.

>15/
>(transport by) ship
>(transport by) plane
>(transport by) rail

The second is different since it involves air transportation, while the other two involve modes on the ground.

>16/
>Military troop strength in <InternationalAgent>
>Military presence in <InternationalAgent>
>Military readiness in <InternationalAgent>

The third is different since it involves a measure of a military force's ability to fight, while the other two discuss some aspect of the numbers of troops.

>17/
>Military security in <InternationalAgent>
>(Security of) <Force>
>Deterrence against <MilitaryAction> by <InternationalAgent>

The third is different since it is a component of the other two--in other words, deterrence is a way of achieving security.

>18/
>Enhance capabilities (of criminal group)
>Expand operations (of criminal group)
>Expand group size (of criminal group)

Again, the third is different since it is a means or maybe more appropriately a requirement to meet the other two selections, which seem to be ends.

>19/
>Earn profits (for criminal group)
>Maintain security (of criminal group)
>Increase prestige (of criminal group)

The first is different since it involves a tangible item, money, to untangibles like prestige and security.

>20/

>6/
>Weaponize weapons of mass destruction
>Demonstrate capability to use weapons fo mass destruction
>Attack <InternationalAgent> using weapons of mass destruction

The first action is one step in the process of acquiring a WMD capability. The other two involve the actual use of WMD.

>7/
>Experience a civil war
>Experience a persecution of <InternationalAgent>
>Experience a domestic man-made disaster

It isn't clear what "experience a persecution of <Int'lAgent>" or "domestic man-made disaster" mean. Since states are not subject to persecution, one can assume that persecution refers to discriminatory/repressive actions taken against subnational agents, such as minority or religious groups. Man-made disaster could refer to an environmental/health disaster resulting from something like an industrial accident. One interpretation: "experience civil war" and "experience persecution" are actions carried out intentionally by some agent/set of agents. Man-made disasters generally are not intentional (but could be).

>8/
>Hold an election for <PersonType>
>Carry out a revolution against <InternationalAgent>
>Carry out a persecution of <InternationalAgent>

The second and third are negative and potentially violent actions directed at some actor (e.g., gov't, minority group). The first is not.

>9/
>Smuggle <ProductType>
>Steal <ProductType>
>Steal classified information regarding <InternationalAgent>

Smuggling involves the illegal transportation of a product. The second two actions are basic theft.

>10/
>Managing public health
>Deterrence against genocide
>Security of citizens

If "security of citizens" is intended to be an interest (which isn't clear since unlike the other two it has no verb attached), then it and the first item can be viewed as domestic interests whereas "deterrence against genocide" (whatever that means) can be viewed as a foreign policy interest. (At the same time, domestic actors can seek to deter genocide in their own country).

>11/
>Economic dominance in <InternationalAgent>
>Cultural influence in <InternationalAgent>
>Military credibility in <InternationalAgent>

These constructions really only make sense if <InternationalAgent> is a <GeographicalRegion>. Otherwise, they should say "of" rather than "in." If this is what they mean, then economic dominance and military credibility depend upon concrete indicators such as the size of a country's economy, the scope of its trade relations, the size of its armed forces, the quality of its weapons, its history of military engagements. Cultural influence depends upon the extent to which countries/societies adopt a foreign culture/cultural symbols/mores.

>12/
>Separatism of military Special Forces
>Secession of <GeographicalRegion>
>Irredentism of military legislature

"Military legislature" doesn't make any sense.

>13/
>Regional promotion of religion
>Territorial claim to (regional) <GeographicRegion>
>Dispute regarding <InternationalAgent>'s policy to develop (regional)
><GeographicRegion>

The second and third imply a specific type of dispute. An actor's efforts to promote religious ideas do not necessarily lead to a dispute with other actors.

>14/
>Rights to <ProductType>
>Infrastructure for <ProductType>
>Price of <ProductType>

The first two items are inherent to the ownership and production of <ProductType> -- a process controlled by a particular agent. The third is dependent on market forces unless it is subject to state control.

>15/
>(transport by) ship
>(transport by) plane
>(transport by) rail

The plane involves aerial transport. The other two modes of transportation occur on the earth's surface.
>
>16/
>Military troop strength in <InternationalAgent>
>Military presence in <InternationalAgent>
>Military readiness in <InternationalAgent>

Troop strength and presence are practically synonymous. Readiness denotes the training status of a military force.

>17/
>Military security in <InternationalAgent>
>(Security of) <Force>
>Deterrence against <MilitaryAction> by <InternationalAgent>
>
The first two denote the status of something. The third implies a the use of a threat to dissuade an <Agent> from undertaking an action.

>18/
>Enhance capabilities (of criminal group)
>Expand operations (of criminal group)
>Expand group size (of criminal group)

The second and third denote an increase of some sort (size/scope). The first refers may or may not -- can enhance capabilities qualitatively or quantitatively.

>19/
>
>Earn profits (for criminal group)
>Maintain security (of criminal group)
>Increase prestige (of criminal group)

The first two are central goals of any criminal organization. The last may or may not be.

>20/
>Shoot (specific) <PersonType>
>Blow up (small group of) <PersonType>
>Fatally infect (large subset of) <PersonType>

The first two involve the use of conventional weapons/munitions. The third involves the use of biological weapons (unless the "agent" doing the infecting is a plague/epidemic, in which case the third would be different in that it would not involve the efforts of a human agent to inflict harm on others)

Michael Schenaker:

1. "Diplomatically support ..." is different because there are no tangibles being invested (i.e., "talk is cheap").

2. "Explicitly threaten..." is different because it implies a threatening action by the U.S. against someone, while the other two are attempts to have someone else perform the action.

3. "Initiate managing public health care..." is different because it involves the U.S. establishing someone's infrastructure, and not just remotely injecting money or equipment.

4. "Charge fee..." is different because it is a tariff, and not an effort to slow or stop goods from reaching someone (e.g., embargo).

5. "Conduct Peacekeeping mission" is different because it connotes a sustained effort to execute a long term plan, instead of a singular, unique, and usually rapid action like the other two examples.

6. "Attack..." is different because it connotes the actual versus implied use of WMD.

7. "Experience a persecution of ..." is different because it implies an external aggressor, while the other two are internal events.

8. "Hold an Election..." is different because it is designed to strengthen or maintain a political system, while the other two examples are ways to tear down the current political system.

9. "Smuggle..." is different because it is not stealing.

10. "Managing public health" is different because it is a long term, quality of life issue to the citizenry while the other examples provide immediate, physical security.

11. "Military credibility..." is different because it is internal to the country, while the other two examples are external impacts upon that country.

12. "Irredentism..." is different because it emphasizes taking back territory to add to the whole, versus a separatist mindset which will reduce the existing whole.

13. "Regional promotion of religion" is different because it changes the indigenous people's entire way of life, instead of just a definition of ownership to the land.

14. "Price of..." is different because it is market driven and therefore, conditional.

15. "(Transport) by rail" is different because it has well defined and tightly constrained routes, while sea and air routes are less constrained.

16. "Military Readiness..." is different because it implies presence and denotes the status of forces, while the other two examples explicitly represent the presence of military forces.

17. "Deterrence against..." is different because it can achieve security through many means, while the other two examples rely upon the military to achieve security solely by themselves.

18. "Expand operations ..." is different because it is within the current capability of the group, while the other two increase the capability of the group.

19. "Increase prestige..." is different because it is not essential to the group's survival.

20. "Fatally infect..." is different because its killing mechanism has been outlawed, instead of more traditional and accepted munitions which use blast, frag, and heat mechanisms.

v/r,

Mike

Knowledge Acquisition for Crisis Management: Interests and Actions

John Kingston, AIAI, University of Edinburgh

I have performed a knowledge acquisition "experiment" in which the SMEs were given sets of three actions or interests (derived from the CM CP grammar), and stated which of each was different from the others, and why. The purpose of this experiment was to perform some knowledge acquisition on interests and actions for the purposes of the Y2 CM CP. This document contains the following information, based on the responses to the questionnaire, on subsequent reading, and on work on a previous DARPA project:

- Ways in which interests and actions interact in answering CP questions
- Attributes of actions which contribute to escalation
- Categorization of interests according to derived attributes
- How categories of interests affect other categories of interests

This document is being distributed for comments on
- Whether the categories proposed are meaningful
- Whether the analyses performed provide useful input to the Y2 CM CP
- Whether the categorisations I have ventured are accurate.

How interests and actions interact

The CM CP questions require the following information about interests and actions:

- SQ 202, 209, 219, 220: Analytical factors needed for action options(e.g. risks, motivations)
- SQ 203, 204, 211, 212, 213: Effect of actions on interests
- SQ 250: Effects of actions on other actions by Analytical Factor (e.g. motivation, cause)
- SQ 210: Action1 is an escalation of Action2
- SQ 214: How will Action1 on Interest1 affect Interest2
- SQ 216: What are the components of an interest
- SQ 217, 239: Effects of interests on other interests
- SQ 220: Attributes of Actions (violent/non-violent)
- SQ 223: What Interests are relevant to a situation
- SQ 228, 238, 240: Distinguishing features of interests for different actors
- SQ 236, 237: What interests motivate actions

In summary, the information needed is:
- How interests affect other interests;
- How interests motivate actions;
- How actions motivate other actions;
- How actions affect interests;
- Components of an interest;
- Attributes of interests;
- Attributes of actions.

In the rest of this document, I will focus on the following information:
- Attributes of actions which contribute to escalation;
- How attributes of interests can be used to categorize interests;
- Which categories of interests are affected by actions.

Attributes of actions which contribute to escalation

Deciding what actions are an escalation of other actions is an important piece of knowledge in answering PQs such as 202, 209, 210, and others. Herman Kahn's book "On Escalation", published in the 1960s, was suggested by the SMEs. Kahn gives a 44-step "escalation ladder" ranging from unfriendly words to unrestricted nuclear war. I have chosen to revise Kahn's ladder into a set of statements of the form "actions of type 1 are escalations of actions of type 2". The primary reason for this was that it's difficult to create a sequential ladder when it is unclear which parameters constitute a "bigger" escalation; for example, illegal actions are considered an escalation of legal actions, and actions in the heart of a country are considered an escalation of actions elsewhere in the country, but it's hard to decide whether legal actions in the heart of a country are higher or lower in the escalation scale than illegal actions elsewhere in the country. The second reason for revising Kahn's ladder of escalation is that actions involving nuclear weapons started appearing about halfway up the ladder; in my view, 90s thinking is sufficiently different from the thinking of Kahn's time (just after the Cuban missile crisis) that the use of nuclear weapons and other WMD should be considered very high on the escalation scale.

My suggested framework is given below. The notation *action1 < action2* implies that *action2* is an escalation of *action1*.

Non-damaging (e.g. verbal) actions < actions damaging terrain < actions damaging property < actions damaging population

Military nearby < military in theater < military in action

Target-specific conventional weapons (e.g. rifles) < target-indiscriminate conventional weapons (e.g. saturation bombing) < weapons of mass destruction

Actions outside target country < actions inside target country < actions inside "heart" (e.g. capital) of target country

Localised actions < widespread actions

Legal actions (by the Geneva Convention, or whatever) < illegal actions

Actions by proxy (e.g. terrorist group) < actions by aggressor nation

Actions reducing flow of supplies < actions cutting off flow of supplies

Short term actions < long term involvement

I have also come across a Web page that describes three models of escalation behaviour which I think could be very useful: http://spot.colorado.edu/~wehr/40RD5.HTM

Categorization of interests according to derived attributes

The SMEs' answers to the questions I posed (see Appendix A) suggested a number of ways in which actions (questions 1-10) and interests (questions 11-20) differed from each other. I have categorized the differences between interests into four groups:

- Credibility vs. Power. The success of some interests requires winning the trust of <InternationalAgent> by capturing and changing the hearts and minds of the people. Other interests are better served by having increased power or authority over <InternationalAgent>, so that the people can, if necessary, be coerced into supporting that interest.

- Tangible vs. Intangible. Some interest involve tangible objects – physical infrastructure, materiel, hardware, physical assets. Others involve intangibles – rights to do something, prices, relationships.
- Quantity vs. Quality. Some interests are concerned with making something bigger (e.g. increasing military presence in <InternationalAgent>). Others are concerned with making something better (e.g. increased military readiness in <InternationalAgent> implies a better ability to act quickly). The grammar for "Interest Effect Types" reflects this; verbs included in this grammar include "strengthening" and "weakening" (in this context, these verbs are primarily quantity-related) and "improvement" and "degradation" (quality-related verbs).
- Fixed vs. variable. Some interests are fixed and unchangeable, at least over the time-scales of a crisis; infrastructure is a good example. Other interests fluctuate more frequently, such as prices or economic indicators. N.B. Fixed interests usually *can* be altered during the course of a crisis (e.g. by military destruction, or by major construction programs such as the supply road to Jerusalem built during a 3-week cease-fire in one of the Middle Eastern wars), but this requires considerable effort.

The table below categorizes all interests listed in the Y2 CM CP spec. on the above four dimensions. The legend is as follows:

C:	Credibility
P:	Power
T:	Tangible
I:	Intangible
N:	Quantity
L:	Quality
F:	Fixed
V:	Variable
-:	not applicable
?:	unsure

The "Who is affected" column draws on Col. R. Worden III's "five rings" theory which is used to determine targets in air campaign planning [Worden, 1996]. These "five rings" are:

- Leadership
- Key Production
- Infrastructure
- National Population
- Fielded Forces

I have drawn on AIAI's previous work on a project for modelling the air campaign planning process, and on information collected during knowledge acquisition for the COA challenge problem, to produce the following categories:

- Leadership
  - Political leadership
  - Military leadership
  - Intelligence capability
- Key Production
  - Electricity
  - Petroleum
  - Chemical
  - Military vehicles
  - Military munitions
  - Raw materials
  - Civilian goods
- Infrastructure
  - Transportation
    - Road
    - Rail

        - Civil Air
        - Inland waterways
        - Coastal shipping
    - Communications
        - Telephone
            - Landlines
            - Mobile phones
        - Computer networks
        - Satellite
        - Vehicle
        - TV/radio
        - Newspapers/printed media
    - Fuel
        - Gasoline
        - Aviation fuel
        - Other
    - Power
        - Electricity distribution
- National Population
    - Economy
    - Health
    - Confidence
    - Culture/Religion
- Fielded Forces
    - Land forces
        - Mobile
        - Static
    - Naval forces
        - Mobile
        - Static
    - Air forces
        - Mobile
        - Static
    - Special weapons
    - Logistics
        - Land supply
        - Naval supply
        - Air supply
        - Storage
        - Repair facilities

	Who is affected	Credib-ility vs. Power	Tangibles vs. Intangibles	Quantity (N) vs. quality (L)	Fixed variab
Economic Interests:					
<EconomicSector>	Population: Economy & Production: all relevant types	C	I	-	F
<ProductType> [{facility, infrastructure}]	Infrastructure:	-	T	Both	F

	<ProductType>				
<EconomicMeasure>	?	-	I	L	V
[construction of] <OilInfrastructureThingType>	Infrastructure: Fuel	-	T	-	F
<TransportMode>	Infastructure: Communic-ations: Vehicle		T (vehicles)	Both	F
<TransportRoute>	Infrastructure: Transportation		Either (road/rail is T, air/sea/cross-country is I)??	Both?	V
markets [for <ProductType>]	Population: Economy		I	Both	V
business connections with [members of] <InternationalAgent>	Population: Economy & Population: Confidence	C	I	L	V
[{resources, right}] [to develop <InterestType>]	Population: Economy & Leadership: Political Leadership	P	I	Both	V
Political Interests:					
<InternationalAgent>	All		?		F
{[{friendly, antagonistic}] [<InternationalCategory>] relations, tensions, dispute} [regarding <Proposition>] {with <InternationalAgent>, between <InternationalAgent1> and <InternationalAgent2>}	Leadership: Political	C (the interest is relations, not what is disputed)	I	L	V
responsibility over the holy sites of Islam	Population: Culture/ Religion	P?	I	N?	F?
{domestic, regional, global} promotion of {ideology, religion, human rights, public health}	Population: Culture/ Religion	C (P? on human rights)	I	?	V?
territorial {sovereignty, claim to <GeographicalRegion>}	Leadership: Political & Population: Culture/ Religion	C	I	L	F
{separatism, irredentism, secession} of <GeographicalRegion>	Leadership: Political & Population: Culture/ Religion	C in region, P outside region	I	L?	F
<GovernmentSector>	Leadership: Political	Both?	?	L, and maybe N in a socialist government	
Military Interests:					
military {forces and capabilities, presence, readiness,	Fielded Forces	Both	I	Both	V

troop strength} in <InternationalAgent>			(readiness) T (all others)		
ability to respond militarily to a crisis in <InternationalAgent>	Fielded Forces	P	I	L or maybe Q	F?
{security, defense} of <InternationalAgent>	Fielded Forces & Leadership: Intelligence & Population: Confidence	P	I (though defensive military assets may be T)	L?	F?
<Force>	Fielded Forces	P	T	L and maybe N	F
[deterrence against] <EventSpec>}	Fielded Forces & leadership: Military & Leadership: Intelligence	P	I	?	F
Criminal Interests:					
earn profits	Population: Economy (N.B. "population" here refers to the criminal group)	-	I	N (of course ☺)	-
maintain security	Leadership: Intelligence	P	I	L	-
expand group size	Fielded Forces	Both	I	N	-
increase prestige and influence	Leadership: Military	Both	I	?	-
expand operations	Production: Goods	P	I?	N	-
monopolize sectors of criminal activity	Leadership: Political & Population: Economy	P	I	?	-
enhance capabilities or resources	Production: Raw Materials	P	I	L	-
Other interests:					
deterrence against [{domestic, regional, global, transnational}] {[proliferation of] <Force>, terrorism, genocide, crime, narcotics trafficking}	Leadership: Intelligence & Fielded Forces (including police etc.)	P	I	Both	F
managing {immigration and emigration, population growth, urbanization, development, public health}	Leadership: Political, Population: Economy, Population: Health	P?	I	L	F
{well-being, security} of citizens [{residing, traveling}abroad]	Population: Confidence	P	I	L	F
[{control, domination, annexation} of] {<TerritoryType>, <InternationalAgent>}	Fielded Forces & Leadership: Political & Leadership: Military	P	I	Both	?

[<InternationalCategory>] [{domestic, regional, international}] {standing, credibility, influence, leadership, dominance} [in <InternationalAgent>]	Leadership: Political	C	I	L?	F?
[{domestic, regional, international}] [<InternationalCategory>] {stability, instability, unrest, tension} [of <InternationalAgent>]	Leadership: Political	?	I	?	F?
{diplomatic, military, economic} commitments to <InternationalAgent>}	Leadership: Political & Population: Economy & Fielded Forces	P	I	???	F

It is clear that not all of the categorizations apply to all the interests.

How categories of interests affect other interests

This table indicates how categories of interests can affect other interests. The table should be read thus: if the contents of a cell are X, then disrupting/improving the interest on the left of this row will disrupt/improve X (where X is a subcategory of the interest at the top of the column).

	Leadership	**Production**	**Infrastructure**	**Population**	**Fielded Forces**
Leadership:					
Political	Military, maybe intelligence	All aspects	All aspects (slowly)	Economy, confidence	All aspects (slowly)
Military	Political, intelligence		Transportation, communication, fuel	Confidence	All aspects
Intelligence Capability				Confidence	
Key Production:					
Electricity		All aspects	Communication (all types)	Economy	
Petroleum		Chemical, maybe electricity	Communication : vehicles	Economy	All aspects except storage
Chemical		Civilian goods, military munitions		Economy	
Military vehicles	Military			Economy	All aspects
Military munitions	Military			Economy	All aspects
Raw materials		All types		Economy	
Civilian goods			Communication : telephones, computer networks, TV & radio, newspapers & print media	Economy	
Infrastructure:					
Infra: Transportation:					
Road	Political	All types	Communication : vehicles,	Economy, confidence,	Mobile land forces,

			newspapers & print media	health	logistics: land supply
Rail	Political	All types	Communication : vehicles, newspapers & print media	Economy, confidence, health	Mobile land forces, logistics: land supply
Civil air	Political	Civilian goods, any other high value-to-bulk production	Communication : vehicles, newspapers & print media	Economy, confidence, health	
Inland waterways	Political	All types	Communication : vehicles, newspapers & print media	Economy, confidence, health	
Coastal shipping		All types	Communication : vehicles, newspapers & print media	Economy, confidence, health	Logistics: naval supply?
Infra: Communications:					
Telephone (landline)	All types	All types (affects good management)		Economy, confidence	All types (affects good management)
Mobile telephone	All types	All types (affects good management)		Economy, confidence	All types (affects good management)
Computer network	Intelligence?	All types (affects good management)		Economy	All types (affects good management)
Satellite	All types	All types (affects good management)		Economy	All types
Vehicle	All types	All types (affects good management)		Economy, confidence, health	All types
TV/radio	Political			Confidence	All types that use radio communications
Newspapers/print media	Political			Confidence	
Infra: Fuel:					
Gasoline		All types (staff cannot get to work)	Communications: vehicle	Economy	Mobile land forces, mobile naval forces, logistics land and naval supply
Aviation fuel				Economy	Air forces, logistics air supply

Other fuel		Maybe Electricity	Communic-ations: vehicle		
Infra: Power					
Electricity distribution		All types	Communic-ations: all types except vehicles	Economy	Static forces (radar stations etc.)
Population:					
Economy	Political	All types		Confidence	Long term effects only
Health	Political			Confidence	
Confidence	Political			Economy	
Culture/ Religion	Political			Confidence	
Fielded forces:					
Land forces	Military	Military vehicles, military munitions	Transportation and communication: expedient construction or repairs	Confidence	
Naval forces	Military	Military vehicles. Military munitions		Confidence	
Air forces	Military	Military vehicles, military munitions		Confidence	Land forces (less air cover)
Special weapons	Military	Chemical, military munitions		Confidence (big effect), health (especially biological weapons)	
FF: Logistics:					
Land supply		Military vehicles	Construction of military roads etc.		More effective
Naval supply		Military vehicles			More effective
Air supply		Military vehicles			More effective
Storage				Confidence? Health? (nuclear storage)	Improves logistics
Repair facilities					All types

Appendix 1: SMEs' responses to questions

John Picarelli

>1/
>Economically support <InternationalAgent>
>Militarily support <InternationalAgent>
>Diplomatically support <InternationalAgent>

The third is different since the first two likely involve the transfer of a physical item to the agent, while the last would only involve words.

>2/
>Explicitly threaten <InternationalAgent> with <EventSpec>
>Demand <EventSpec> of <InternationalAgent>
>Attempt to intimidate <InternationalAgent> into <EventSpec>

The third likely involves an indirect approach, while the other two seem pretty direct.

>3/
>Increase economic aid to <InternationalAgent>
>Provide military aid to <InternationalAgent>
>Initiate managing public health for <InternationalAgent>

The third is different since it involves one type of activity, managing public health, while the other two involve a different activity, dispersing international aid of some sort.

>4/
>Embargo <ProductType> to <InternationalAgent>
>Curtail flow of <ProductType> to <InternationalAgent>
>Charge fee for use of transport for <ProductType>

The first is different since it involves the severing of trade flows, while the other two serve to hamper trade flows.

>5/
>Conduct peacekeeping mission
>Conduct search and rescue mission
>Conduct counter-terrorism mission

The middle involves a mission where the military will not use force, while the other two imply the use of force.

>6/
>Weaponize weapons of mass destruction
>Demonstrate capability to use weapons fo mass destruction
>Attack <InternationalAgent> using weapons of mass destruction

The third is different since it involves the employment of a WMD, while the other two only involve the development of a WMD.

>7/
>Experience a civil war
>Experience a persecution of <InternationalAgent>

>Experience a domestic man-made disaster

The first is different since it involves war, while the other two do not.

>8/
>Hold an election for <PersonType>
>Carry out a revolution against <InternationalAgent>
>Carry out a persecution of <InternationalAgent>

The first is different since it involves a civil process, while the other two involve illicit actions within a state.

>9/
>Smuggle <ProductType>
>Steal <ProductType>
>Steal classified information regarding <InternationalAgent>

The first is different since it involves one type of illicit activity, smuggling, while the other two involve a different type of illicit activity, theft.

>10/
>Managing public health
>Deterrence against genocide
>Security of citizens

The second is different since it is a foreign policy, while the other two are likely domestic policies.

>11/
>Economic dominance in <InternationalAgent>
>Cultural influence in <InternationalAgent>
>Military credibility in <InternationalAgent>

The third is different since it involves one type of interest, creadibility or trust, while the other two involve another type of interest, power (in this case, the ability to influence or control the actions of others).

>12/
>Separatism of military Special Forces
>Secession of <GeographicalRegion>
>Irredentism of military legislature

[John, I am unsure about what a 'military legislature' is]. The third is different since it involves the separation of a geographic region while the other two involve a military body.

>13/
>Regional promotion of religion
>Territorial claim to (regional) <GeographicRegion>
>Dispute regarding <InternationalAgent>'s policy to develop (regional)
><GeographicRegion>

The first is different since it involves the promotion of an ideology,

Knowledge Representation and Reasoning: Problem Solving Methods

These appendices summarises the results from Work Package 6. This work package originally set out to build a generic library of problem solving methods, in conjunction with a problem solving methods working group that was set up with other HPKB participants. However, it turned out that problem solving methods were not necessary to solve the Challenge problems, and as interest in PSMs waned, AIAI scaled down its efforts on this work package. However, enough groundwork was done to greatly simplify the development of a libvrary of problem solving methods.

This groundwork is described in the following three deliverables:

- Report comparing two approaches to knowledge engineering: EXPECT and KADS.
- Report on capability descriptions for problem solving methods.
- Implementation of a problem solving method in Cyc.

E Comparison of EXPECT and KADS

Stuart Aitken

HPKB Report, August 1997

Abstract

This working paper compares the approaches to knowledge acquisition, knowledge modelling and KBS maintenance proposed in the KADS and EXPECT methodologies. Knowledge representation languages and tool support are also examined.

E.1 EXPECT: A Brief Summary

The EXPECT project [4, 5, 7, 14] is primarily concerned with the problem of modifying existing knowledge bases. The EXPECT tools support the user in the task of updating the KBS, both at the level of adding new domain knowledge, and at the level of modifying the problem-solving methods employed. The EXPECT project claims to take an explicit approach to knowledge representation, to be more adaptable than the role-limiting method approaches, and to be supportive to the user.

EXPECT uses LOOM for domain knowledge representation and has an integrated language for representing method knowledge [4]. Methods are defined in a typed language which is used to specify the goal, result and method-body slots of a problem-solving method definition. These languages play an important role in EXPECT; the user is expected to understand and modify definition and program expressions in order to extend the knowledge base and to modify a problem-solving method. Further, statements in the languages are analysed by the system's tools to check for errors in syntax and semantics.

An EXPECT KBS could be designed, or re-designed, using structured methods and EXPECT does have the notion of roles for knowledge items. There is no clear notion of generic methods. In fact, the user is free to code or re-code a problem-solving method arbitrarily. Re-coding often requires several modifications to the knowledge base and EXPECT supports the user by suggesting a knowledge acquisition (KA) Script [7] which is a stereotypical sequence of steps. Maintaining knowledge base structure is not supported above the implementation level.

EXPECT supports what might be termed the *evolution* of the knowledge base during its lifetime – a task which clearly requires the acquisition of knowledge. Maintenance would be the more conventional term for this phase of the KBS life-cycle. EXPECT does not address the initial analysis, modelling and design phases of KBS development.

E.2 A Comparison of EXPECT and KADS

EXPECT and KADS address problems which occur in different phases of the KBS life-cycle. Nonetheless, it is possible to compare the approaches.

E.3 Knowledge Acquisition and Knowledge Modelling

Knowledge acquisition is supported in KADS by the use of inference structures to focus the KA effort. These models serve as reference points for the knowledge engineer and are not intended to be as prescriptive as role-limiting methods (KADS advocates the principle of differentiating simple models into more complex ones). Therefore in KADS, KA is guided and methods are flexible – there is a strong emphasis on analysis and the structured approach is maintained by respecting the semantics of the predefined knowledge sources and maintaining the domain/inference/task level distinction. Other principles of the KADS methodology include: the use of multiple models to cope with complexity, the re-usability of generic model components, and the importance of structure preserving design [16].

As stated above, KA and KBS design in EXPECT could be carried out in a structured way, but this is not enforced. EXPECT does not make an essential distinction between domain-specific and method knowledge - these are viewed as points on a continuum [5]. Such a distinction has been the basis of most structured approaches (see [2]). The case for abandoning this principle appears to be made on usability criteria and not on epistemic criteria.

The advantages of the structured approach have been convincingly demonstrated by Kingston *et. al* [8] in complex applications such as planning, as well as in more conventional expert systems domains.

E.4 Knowledge Representation

EXPECT claims an explicit approach to knowledge representation, this is based on the use of an interpreted knowledge representation language (KRL) for both domain and problem-solving knowledge. The approach is explicit in the sense that all textual languages are explicit, but, at the method language level, it cannot be said to be be declarative, or logically formal. A number of formal languages for expressing KADS models have been developed. The claimed advantages are: the improvement of clarity of specifications, that completeness and consistency can be verified, and that formal specifications can be mapped to operational ones [3]. Similar arguments can be made for formalising the domain ontology [15]. We now discuss the knowledge representation languages in detail.

E.4.1 Domain Knowledge Representation in EXPECT

EXPECT uses LOOM to represent domain knowledge. LOOM is itself a programming language and environment for constructing knowledge-based systems [9]. It is implemented in Lisp. LOOM allows users to define concepts and relations, query the knowledge base, define methods (in an object-oriented style) and rules (in a production-rule style). LOOM has a *classifier* which automatically deduces the subsumption relation between concepts in the knowledge base and also does constraint reasoning automatically.

In LOOM, *concepts* name a class of entities. Concepts can be primitive or defined, and can have constraints associated with them. For example, *Robot* can be defined to be a primitive concept, *FactoryRobot* a defined concept as follows:

```
(defconcept Robot)

(defconcept FactoryRobot
     :is (:and Robot (:exactly 2 robot-arms)))
```

The distinction between primitive and defined concepts is that an instance I which has all the properties of a defined concept D can be automatically deduced to be of class D, while I would not be deduced to be an instance of a primitive class P, where P is otherwise equivalent to D, as primitive classes are presumed to have unspecified properties - which I may not have.

A concept definition can include constraints:

```
(defconcept Physical-Object
     :constraints
         (:and (:exactly 1 weight}
               (:exactly 1 location)))
```

Constraints are composed of *concept-expressions* [9] which include logical connectives and quantitative expressions, e.g. *exactly*. Concept-expressions may refer to relations, e.g. *location* in the above example.

Relations are defined to hold from a specified domain to a specified range, both of which may be concepts, and can have the characteristic of being closed-world or open-world [10]. A relation which is closed-world is assumed to hold of only those items to which it is currently known to hold. This characteristic affects LOOM's automated reasoning about relations. An example of a relation is given below.

```
(defrelation weight-in-kilos
     :domain Physical-Object
     :range  Weight-in-kilos
     :attribute :single-valued)
```

Weight-in-kilos must be defined – as a concept defined to be a number between zero and positive infinity.

Concepts and relations are the basic KR primitives. LOOM is capable of reasoning about concept subsumption relations and constraints automatically. It should also be mentioned that LOOM has been extended to provide new features which permit temporal assertions to be made and which automate temporal reasoning. The programming constructs provided by LOOM are not relevant here as they are replaced by EXPECT's method language, which we discuss later.

E.4.2 Domain Knowledge Representation in KADS

In KADS, domain knowledge may be represented in CML [12, 17], although many other formal and operational languages have been defined [3]. The primitives of CML are concepts, attributes, expressions and relations. CML is a notation and not an executable language.

Concepts name classes of objects and instances of concepts may have *attributes* associated with them. `Component` is an example of a concept; instances of this concept are part types of an artifact, e.g. car and elevator in the VT domain. An `attribute-slot` is an example of an attribute; instances of this attribute include weight and height. Value sets define the range an attribute may take and form part of the attribute specification.

Expressions specify the range of values of an attribute or property. For example, `height = 28.75` is an expression which specifies the value of the attribute slot height. Expressions are associated with instances of concepts.

Relations link concepts, attributes, expressions and relations. The relation `has-model` holds between two concepts and is defined as follows:

```
Relation: has-model
   argument-1: component
   argument-2: component-model
```

The relation `has-attribute` holds between a concept and an attribute:

```
Relation: has-attribute
   argument-1: component \/ component-model
   argument-2: attribute-slot
   axiom:      has-attribute(c,a) /\ has-model(c,m)
               ==> has-attribute(m,a)
```

The definitions given above form part of the *model ontology* for the VT problem. The model ontology provides the meta terms that describe the domain. The *domain ontology* differs from the model ontology as it need not reflect any conceptualisations which arise from the use of a specific problem-solving method. If there is an existing knowledge base then its ontology will probably differ from the required model ontology and it will be necessary to define a mapping from the domain to the model ontology. These issues are important in knowledge re-use.

The notion of *concept* is superficially similar in LOOM and CML. However, in KADS/CML conceptualisations are related to problem-solving activity (the model ontology) or to the domain (the domain ontology) while in EXPECT/LOOM these two possible views of a domain are not distinguished.

Constraints in LOOM could probably be modelled by expressions in CML, and relations are similar in both languages (except that in CML relations can be defined to hold between relations). Some constraints would probably have to be modelled by domain rules in CML. Reasoning about constraints and subsumption is handled automatically in LOOM, but would be explicitly reflected at the inference layer in CML. We now consider method knowledge in EXPECT and CML.

E.4.3 Method Knowledge Representation in EXPECT

Methods in EXPECT are user-defined programs which make reference to domain knowledge and to other methods. The approach is similar approach to that taken by LOOM. Methods have the following structure:

```
(defmethod REVISE-CS-STATE
     ''To revise a CS state, apply the fixes ...''
     :goal         <pattern-1>
     :result       <pattern-2>
     :method-body  <pattern-3>)
```

The patterns are s-expressions: a method is applicable if the goal pattern matches the current goal, the result pattern specifies the type of result, and the result is calculated by evaluating the method-body pattern. The explicit statements of the goal and result types allows EXPECT's tools to analyse the knowledge base for omissions and errors.

The applicability of a method can be generalised by stating that the goal applies to any specialisation of a super-class of concepts, and not just to a specific concept [13]. Data abstraction can also be used to create more re-usable methods. An additional feature of EXPECT which increases the flexibility of the use of methods is goal reformulation: In certain circumstances, EXPECT can reformulate a goal to create a new goal decomposition of subgoals which the system is able to solve.

Methods can be domain specific or domain independent - as the user sees appropriate. The implementation of a method is also the specification; these properties are not distinguished. Consequently, changing the implementation may change the conceptual view of how the method operates, but the significance of such changes may go unnoticed.

The EXPECT approach could be extended to advise the user on knowledge structuring, e.g., new methods could be analysed for re-usability and ontologies could be analysed. The approach is not incompatible with KADS, but its applicability would depend on whether the KADS languages could be analysed to the necessary extent.

E.4.4 Method Knowledge Representation in CML

Control knowledge in CML is represented at two levels: the inference level and the task level. The network of inferences that make up a problem-solving method is called the inference structure. Each inference is specified by its name, and the input and output concepts, or sets of concepts (termed roles). The inference structure is often drawn as a data flow diagram. The inference `select-parameter` is defined as follows:

```
inference  select-parameter
```

```
operation-type: select
input roles:
   parameter-set --> set of attribute-slots
   parameter-assignments --> set of tuples
                  <attribute-slot,value,dependencies>
output roles:
   parameter --> single attribute-slot
static roles:
   formulae in domain models initial-values
   and computations
spec:
    ''Select a parameter ...''
```

This specification defines the mappings from domain to method ontology for both input and output roles, and states which domain models (partitions of the knowledge base) are used in the inference. No details about the implementation of the inference are given.

Task knowledge describes how a goal can be achieved through a task. Tasks may be composite (referring to other tasks), primitive, or transfer tasks (denoting operations in the outside world). The task level specifies when during problem solving an inference gets applied: tasks include procedures which correspond to inferences.

A *task-definition* describes what needs to be achieved. It is composed of a goal, input/output roles (both of these are textual descriptions) and a task specification - being a description of the logical dependencies between tasks. The *task-body* describes how the goal is achieved. The procedural code which defines the calculation forms part of the task-body. The (pseudo) programming language used in the task body includes the usual assignment, conditional, and repetition operators.

Translating an EXPECT KBS into CML might require extensive analysis to distinguish domain and model ontologies[1]. As CML does not have the notion of constraint, constraint knowledge in LOOM would have to represented in some other way in CML. Further, methods in EXPECT would have to be reconstructed as inference structures and tasks in CML, taking LOOM's automated reasoning into account.

As illustrated, the KADS methodology advocates the use of explicit models for many of the elements of a KBS. The pattern of inferences between concepts is represented by the inference structure. This structure is generic and potentially re-usable. The task decomposition is defined in terms of inferences and is the source of control over inference invocation. Again, the task structure may be re-used. EXPECT, in contrast, does not require the use of explicit, unimplemented, models. Consequently, the re-use of methods would appear to depend on ability to re-use method code. The intertwining of problem specific and problem independent knowledge would also appear to inhibit re-use of method and domain knowledge.

[1]This issue is briefly mentioned by Swartout and Gil [14] in a comparison of knowledge acquisition systems.

E.5 Maintenance

There is little existing analysis on the possible differences between initial KA tasks and knowledge evolution/maintenance tasks. Differing assumptions may be made about who will be performing these tasks (i.e. about their expertise), and how structured approaches can be used.

Maintenance in KADS is relatively unexplored, but it is easy to visualise users extending the domain knowledge of an existing KBS. Specialised tools, similar to those developed for role-limiting methods, could be employed. Modifying the problem-solving method would generally require reviewing the inference and task knowledge and this is unlikely to be an end user activity. In this respect, KADS stands between EXPECT and the role-limiting approaches as problem-solving methods in KADS *can* be refined, but this requires specialist knowledge.

EXPECT permits all types of knowledge to be re-organised by any user who is a competent programmer. No structuring principles are enforced or recommended, but EXPECT's tools check for the consistency and completeness of KB updates and provide sophisticated guidance to the user when modifying the KBS.

While EXPECT is presented as an approach to knowledge acquisition, the proposed techniques address **usability problems** which arise in the extension and modification of the KBS. In EXPECT, tool support is aimed at experienced users and/or programmers, while in in KADS, tool support is aimed at analysts and system builders.

E.6 Conclusions

EXPECT aids the user extend, or evolve, an existing KBS. The tools provide advice which helps to ensure knowledge base consistency. EXPECT provides powerful automation and user support, but lacks a concrete methodology for knowledge acquisition tasks - as is evidenced by the focus on examples in the published work [4, 5, 7, 13, 14]. The EXPECT approach could be characterised as *critiquing* to support knowledge acquisition.

EXPECT does not advocate any explicit principles for method knowledge structuring or re-structuring, but inherits LOOM for domain knowledge representation. It is difficult to assess claims for the generality or re-usability of problem-solving methods from the published examples, and without a clear view of the context in which re-use is to take place. EXPECT emphasises technologies for knowledge acquisition, while KADS is almost exclusively concerned with modelling and methodology: The approaches are complementary. Gil and Paris [6] also note that EXPECT and KADS address tasks in complementary phases of the KBS life-cycle.

References

[1] Aben, M. Formally specifying reusable knowledge model components. *Knowledge Acquisition* 5, 1993, pp. 119–141.

[2] Clancey, W. Heuristic classification. *Artificial Intelligence* 27, 1985, pp. 289–350.

[3] Fensel, D. and van Harmelen, F. A comparison of languages which operationalize and formalize KADS models of expertise. *The Knowledge Engineering Review* Vol. 9:2, 1994, pp. 105–146.

[4] Gil, Y. Knowledge refinement in a reflective architecture. *Proceedings of the Twelfth National Conference of Artificial Intelligence (AAAI-94)*, Seattle, WA, August 1994.

[5] Gil, Y. and Melz, E. Explicit representations of problem-solving strategies to support knowledge acquisition. *Proceedings of the Thirteen National Conference on Artificial Intelligence (AAAI-96)*, Portland, OR, August 4-8, 1996.

[6] Gil, Y. and Paris, C. Towards method-independent knowledge acquisition. *Knowledge Acquisition, Special issue on Machine Learning and Knowledge Acquisition*, Vol. 6(2), 1994, pp. 163–178.

[7] Gil, Y. and Tallis, M. A script-base approach to modifying knowledge bases. *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, Providence, RI, July 27-31, 1997.

[8] Kingston, J., Tate, A., and Shadbolt, N. *CommonKADS models for knowledge based planning* AIAI Technical Report AIAI-TR-199, 1996.

[9] *LOOM users guide* ISX Corporation, August 1991.
URL: `http://www.isi.edu/isd/LOOM/documentation/usersguide1.4.ps`

[10] *The Loom tutorial* Artificial Intelligence research group, Information Sciences Institute, USC, December 1995.
URL: `http://www.isi.edu/isd/LOOM/documentation/tutorial2.1.html`

[11] MacGregor, R.M. Using a description classifier to enhance deductive inference. *Proceedings of the Seventh IEEE Conference on AI Applications*, 1991, pp. 141–147

[12] Schreiber, G., Wielinga, B.J., Akkermans, H., Van de Velde, W., and Anjewierden, A. CML: The CommonKADS conceptual modelling language. *Proceedings of the Eighth European Knowledge Acquisition Workshop*, LNAI 867, Steels, L., Schreiber, G., and Van de Velde, W. (eds.), Springer Verlag 1994, pp. 1–25.

[13] Swartout, B. and Gil, Y. EXPECT: Explicit representations for flexible acquisition. *Proceedings of the Ninth Knowledge Acquisition Workshop*, Banff, Canada, 1995.

[14] Swartout, B. and Gil, Y. Flexible knowledge acquisition through explicit representation of knowledge roles. *1996 AAAI Spring Symposium on Acquisition, Learning, and Demonstration: Automating Tasks for Users*, Stanford, CA, March 1996.

[15] Uschold, M., and Gruninger, M. Ontologies: principles, methods and applications. *The Knowledge Engineering Review* Vol. 12:2, 1996, pp. 93–136.

[16] Wielinga, B.J., Schreiber, T., and Breuker. J.A. KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition* 4, 1992, pp. 5–53.

[17] Wielinga, B.J. (ed.) Expertise model definition document. Technical Report ESPRIT Project P5248, KADS-II/M2/UvA/026.5.0, University of Amsterdam, 1994.

F Capability Descriptions for Problem Solving Methods

Stuart Aitken, Ian Filby, John Kingston, and Austin Tate

HPKB Report, January 1998

Abstract

This paper proposes a set of attributes which specify the capabilities of problem solving methods. This set is intended as a form of documentation for use in a library of problem solving methods (PSMs). The capability attributes we identify provide information to the knowledge engineer to support specific tasks, namely, method selection and method configuration. The view that a capability description should support specific tasks has implications both for the set of relevant attributes, and for the formality of the description.

Previously, PSMs have been described at varying degrees of formality, and for diverse purposes, which include the need to to present and document the conceptualisation of the PSM, and to address more theoretical problems such as to clarify conceptual issues in modelling (e.g by formal proof). Currently, there is interest in the HPKB program in describing PSMs with a view to specifying method configuration and performance in a formal language.

F.1 Introduction

Enabling the reuse of problem solving methods is an important issue in knowledge engineering. PSMs can be described at both conceptual and implementation levels and there is the potential for reuse at both levels. Failure to exploit existing models and code is clearly inefficient, as has been recognised in software engineering, and is a factor which will prohibit the development of high performance knowledge-base systems. The DARPA-funded High Performance Knowledge Bases (HPKB) program is concerned with this problem, and this paper reflects our input to HPKB. Our solution approach is theoretically motivated, taking the experience of recent ESPRIT Knowledge Based Systems (KBS) projects as a starting point. Other HPKB participants are also addressing the problems of PSM library construction [10, 17], but from the more pragmatic perspective of providing users with tool support.

The HPKB program is interested in developing a language for describing PSMs. Considering a language for representing PSMs requires choosing a formalism for representing capability descriptions of PSMs; for this, we need to be clear about who will be using the PSM capability description and for what purpose. The ultimate goal may be the automated configuration of PSMs, but human knowledge engineers will be the initial users (method constructors) and human domain experts may also be users. A variety of languages could therefore be necessary [4]: natural language, a semi-formal representation, and possibly a logical representation. We will therefore avoid the task of suggesting a language to represent our proposed capability attributes until other issues have been resolved.

This paper continues by identifying the strengths and weaknesses of using a planning ontology to document the capability of a PSM. Then, in Section F.3, we identify the knowledge engineering activities that involve the use of capability statements, and begin to outline the relationship between the use of the capability statement and its content. The list of capability attributes we recommend is based on a survey of the literature, which is summarised in Section F.4, and the attributes that we adopt are described in Section F.5. Finally, we discuss related work, and how it may contribute to our approach.

F.2 Problem-Solving Methods as Processes

PSMs are processes, and it could be argued that they can be represented and characterised in the same way as any other process; this suggests that process modelling techniques such as IDEF-3 [11] or planning representations such as the Shared Planning and Activity Representation (SPAR) [15] may be usable for representing PSMs. We believe these representations are useful as a starting point, but do not provide sufficient detail to characterise PSMs fully because PSMs are **knowledge-intensive** computational processes. However, it is important that our representation is consistent with these approaches to enable consistency between our proposed representation of PSMs and more general representations of processes.

Consequently, we begin by instantiating the SPAR description of *process* for PSMs. In SPAR, every process is expected to have:

- an environment, defined in terms of constraints on activities, objects, and time (plus several other attributes);
- an activity specification which is defined in terms of:
  - activity constraints, including
    - resource constraints,
    - actor constraints, and
    - world constraints;

    and
  - sub-activities, each defined in terms of:
    - begin/end time points, and
    - an activity specification (as defined above).

Instantiating the built-in categories for knowledge-intensive PSMs:

- Environment: including computational, interactive and social elements.
- Resource Constraints: including computational resources.
- Actor Constraints: including epistemological and human problem solving resources.
- World Constraints: conditions that must hold for a PSM to be applicable, e.g. data inputs, and effects of executing a PSM, e.g. data output.

- Sub-activities: sub-methods of a PSM, each defined in terms of the above attributes.

As a documentation of a PSM process, the categories listed describe what a PSM does. If the only functional requirement for a capability description of a PSM was to specify the actions of the PSM in the world, then the above documentation scheme would be adequate. But this is not the case (although SPAR offers an ontology which includes Plans, Issues, Agents, Objectives, Evaluation Criteria and Relationships which we could instantiate to say more about processes). We now discuss additional social and computational activities and processes that are involved in the use of a PSM, in order to define requirements for capability descriptions which may go beyond the ontology which SPAR provides.

F.3 Using PSMs: Selection, Configuration, and Execution

Capability descriptions are required for at least three activities. It is first necessary to *select* one or more appropriate PSMs for the task in hand; then it is often necessary to *configure* the PSM to the specific requirements of the domain; and finally it is necessary to *execute* the PSM. Other activities may also occur, for example, knowledge engineers may wish to simulate the execution of the PSM in a planning system, or to formally verify that certain properties hold of a configured PSM, or of its component inferences.

Selection of a PSM may involve properties of a method which are not easily derivable from the representation of the method itself. These include intentions (e.g. the objective), features of the solution (e.g. that it includes an explanation component), the rationale, and other meta-level properties such as optimality, and consistency. Selection may also make reference to the description of the PSM itself; for example, knowledge that a PSM for Repair tasks produces a Diagnosis as an *intermediate output* might be relevant.

Configuration of a generic PSM involves identifying relevant domain theories and mappings, instantiating sub-methods, and ensuring that tasks which must be executed in the environment (non-computational tasks) can actually be executed. Configuration of a PSM may generate executable methods and/or may instantiate generic knowledge roles to domain knowledge.

Execution of a PSM means running the assembly of pre-defined code fragments, or the program designed in configuration—if none existed previously.

A simple schematic representation of these activities is given in Figure 1. In this model, which uses SPAR process attributes, the selection process finds the name of a relevant PSM for a given problem description. This name is input to configuration, which in turn generates an executable PSM-instance, which in turn is viewed as a resource in the execution process.

```
-------------
Process: Selection
Environment: any
Resource Constraints: PSM library exists
Actor Constraints: actor is a knowledge engineer
World Constraints: at begin: problem description exists
                   at end: PSM-name exists
Sub-activities: none
-------------
Process: Configuration
Environment: any
Resource Constraints: PSM library exists
Actor Constraints: actor is a knowledge engineer
World Constraints: at begin: PSM-name exists
                             problem description exists
                   at end: PSM-instance exists
Sub-activities: none
-------------
Process: Execution
Environment: any
Resource Constraints: PSM-instance exists
                      cpu cycles available
Actor Constraints: none
World Constraints: at begin: input data for PSM exists
                   at end: output data exists
Sub-activities: none
-------------
```

Figure 1: A simple model of activities

F.4 Relevant Research

In addition to the SPAR process representation[2] [op. cit.], we have considered a number of approaches to describing capabilities of problem solving methods. These include:

- the design of the CommonKADS library [4, 14, 19];
- CommonKADS' competence theory [2];
- the Components of Expertise approach (Steels [16]);
- the EuroKnowledge ESPRIT project [8, 12];
- the Cokace PSM library tool [6];
- Design patterns for Object-oriented programming [9].

The salient features of each approach are described below:

[2]We use this as the most recent work on a "standard" shared representation of plans, processes and activities. It is representative of similar work on KRSL-Plans, PIF, NIST PSL, OMWG CPR, and O-Plan <I-N-OVA>, see Tate [18].

CommonKADS library The CommonKADS methodology [19] offers a library of "generic descriptions of inference tasks" - in other words, problem solving methods, described at a conceptual (i.e. implementation-independent) level of abstraction. This library can be indexed in two ways: by the *task type* (e.g. diagnosis, assessment, configuration)—an approach familiar to CommonKADS users, and by *solution type*. The task-type approach has been reviewed [5], and the hierarchical view of tasks has been replaced by a set of six *problem types*, of which more than one may be required to solve a particular problem. The six problem types are: modelling, design, assignment, prediction, monitoring, and diagnosis. Breuker [4, 5] has also pointed out that PSMs also need to be distinguished by the components required in a solution: the answer, an explanation of the input data, or an explanation of how the answer was deduced. For diagnostic tasks, this would equate to differentiating between PSMs identifying a faulty component, PSMs identifying a state which caused the fault, or PSMs which justify the link between the input state and the fault.

CommonKADS' competence theory Competence-directed refinement [3] is a formal approach to deriving a PSM from a specification by introducing new elements to the conceptualisation. Methods are described logically, and new predicates are introduced to name significant classes of concepts and domain relations. Different sets of logical statements hold for the different possible refinements of the specification. This approach requires the solution to be formally specified.

Both properties of the solution and properties that hold between elements of the conceptualisation are potentially useful for indexing purposes. Useful properties of the solution include consistency, which may hold between observations, the complaint and the solution, or just between the complaint and the solution. The solution may also be minimal, or optimal.

The assumptions and inter-element properties of a diagnosis method include:

- All complaints have an explanation;
- There is a consistent causal explanation of observations;
- An explanation is a consistent covering of a complaint.

In [3] these statements are specified by axioms, but this level of formality is not essential. Semi-formal or natural language descriptions of competence could be used instead.

Components of Expertise Steels [16] presents a componential framework for KBS construction which involves relating domain models to PSMs and task decompositions. It is argued that selecting a PSM involves both pragmatic and conceptual aspects. Conceptual aspects specify the nature of the input-output relation, while the pragmatic aspect refers to features of the domain knowledge which determine the methods that can be selected, e.g. if domain concepts are defined by typical features, as opposed to necessary and sufficient features, then methods that rely on differentiating concepts by the lack of a feature are ruled out.

A list of attributes which characterise a classification method is presented in an example. These include:

- purpose,
- whether domain features are necessary or prototypical,
- whether the domain theory is uncertain or incomplete,
- whether data collection is system or environment driven, and
- whether knowledge will be accessed in a data or a goal driven way.

The *type* of the domain model and the *method-cliche* are also attribute; however, no generic description of these terms is offered.

EuroKnowledge The EuroKnowledge ESPRIT project surveyed languages for representing conceptual models [8] (including PSMs), and the design of a library of problem solving methods [12]. The conclusions were that it is inevitable that there will be multiple languages for different purposes and that approaches taken in software engineering, and IT in general, were relevant to knowledge engineering. Regarding library design, EuroKnowledge concluded that CommonKADS was the *de facto* standard in Europe.

EuroKnowledge does not appear to offer any new capability definitions, but it reinforces our earlier claim that the choice of a language of expressing PSM capabilities should be considered separately from the identification of the capability description features.

Cokace Cokace [6] is a tool for PSM selection which has a small library of applications and methods. It is based on the CommonKADS view of task, method, and domain knowledge. Cokace is an implemented system with a web-based interface. Its capability features are *task-type* and *application domain*: these are two alternative routes to select a PSM. The list of task-types Cokace presents to the user include reasoning paradigms such as argumentation and case-based reasoning, and problem types, e.g., assessment, design, and diagnosis. After selecting a task-type, the user can choose to look at a list of relevant methods. Using the alternative option, when the user selects an application domain, they are then immediately presented with a menu of methods that are applicable in that domain. For example, two diagnosis methods are accessible after selecting the Car Diagnosis application, while ten methods are accessible under Concurrent Engineering. Only the names of the methods are presented to the user, and these are not always helpful, e.g. Solvel and Solve2 are methods for conflicts solving — a task which is not further described.

Design patterns for object-oriented programming The "design patterns" proposed in [9] for object-oriented programming correspond to PSMs, or components of PSMs, for object-oriented programs. The design patterns are classified according to their *purpose* (object creation, composition of objects, or object interaction) and according to whether they affect classes or instances. A list of features for describing design patterns is also suggested; this list includes several features which appeared in SPAR's recommended list of features for any process (in SPAR's language, these include application conditions, components, and consequences/intents) as well as implementation issues, a graphical representation of the classes involved, and diagrams representing interactions between components.

F.5 A Capability Description for PSMs

We have used the information categories suggested in the literature to define a capability description which supports each of the three processes identified above: selection, configuration and execution. As stated earlier, capability descriptions are viewed as a resource in selection and configuration.

In addition to the model of library use, we have made a number of assumptions about the structure of methods and ontologies. We assume that PSMs can be described at a conceptual level by knowledge roles, that some roles are input to and output from the method, other roles are internal to the method, that method ontologies will generally differ from domain ontologies, and that mappings can be defined between ontologies and between languages. Mappings can be declaratively specified, see [13] for a discussion of the mapping problem. The majority of these (uncontroversial) assumptions are relevant to the configuration process. We have also assumed that knowledge roles and ontologies can simply be named, each name having an accepted meaning. In the case of ontologies, names could be replaced by axiomatic specifications, e.g. in KIF. However, in the example developed later in this section we assume the existence of named ontologies and mappings.

The proposed capability description is subdivided into the sections *Competence*, *Configuration* and *PSM Process* to organise the capability slots into groups which are related to processes[3]. A list of the slots of the proposed PSM capability description can be found in Figure 2. The precise meaning of the terms we use is explained in the remainder of this section, and we also present an example description of a PSM for diagnosis.

Capability Description: Competence

Goal The objective of a PSM. Specifying the objective requires a standard terminology. This is a task feature in CommonKADS.

Problem Type The generic type of problem a method applies to. The set of six problem types identified by Breuker [5] are: { modelling, design, assignment, prediction, monitoring, diagnosis }. Problem Type is a component of task knowledge in CommonKADS. This term refers to problems only, in contrast with the term *task* which is often used as a synonym for method, and combines the notions of 'problem' and 'method of solution'.

Generic Solution Types of solution that can be generated for problems. For example, there are three generic solutions for diagnosis: { set of faulty components, fault classification, causal explanation of fault }. An example of a fault classification in a medical domain is an infection (following Bredeweg [3]). Other problems will add to this set as generic

[3]This organisation of information should not be taken to imply that selection, for example, uses only Competence information.

```
----------------------------------------------------
PSM Capability Statement
Competence (to support selection)
      Goal
      Problem Type
      Generic Solution
      Solution Component
      Solution Properties
      Rationale

Configuration (to support configuration and selection)
      Method Ontology
      Domain Theory Requirements
          Field
          Ontology/Mapping
          Representation
      Sub-methods

PSM Process Description (i.e. a description of the
     execution process to support selection)
      Environment
      Resource Constraints
      Actor Constraints
      World Constraints - Data Input
                        - Data Output
      Sub-activities

Method (a (pseudo) code representation of the method
itself) - this should include a description of the
structure of the method and of interactions between
components.
----------------------------------------------------
```

Figure 2: An indexing scheme for PSMs

solutions are problem specific. Generic Solution is a component of task knowledge in CommonKADS.

Solution Component Three solution components have been identified (Breuker [4]) { conclusion, argument structure, case model }. The argument structure justifies the conclusion (as in a proof), while the case model explains the data. These components are not problem specific.

Solution Properties Solution properties are properties that hold between knowledge roles in the conceptual model of a PSM (some knowledge roles are also inputs and outputs of the method as a whole). Example properties of diagnosis methods are: consistency between complaints and diagnosis, consistency between complaints, observations and diagnosis, minimality of the explanation, and more generally, optimality of the solution. These Solution Properties were identified by Akkermans [2] for CommonKADS models (but were not used as task features).

Rationale The rationale can simply be a textual description of why and when the method might be used. The conceptual model of the method (the inference structure in CommonKADS) could be used to explain the problem solving process. Rationale is a feature of indexing schemes for reusable software components.

Capability Description: Configuration

Method Ontology The name (or specification) of the ontology assumed by the method. Ontology is a feature in the Cokace library tool. Knowledge types are a similar task feature in CommonKADS.

Domain Theory - Field The field of knowledge of a domain theory, e.g. medicine. Application field is a feature in the Cokace library tool.

Domain Theory - Ontology/Mapping The name (or specification) of the domain ontologies which can be mapped to the method ontology.

Domain Theory - Representation The name of the language in which the domain theory is represented.

Sub-methods A specification of sub-methods required in the configuration of this method. Sub-methods can be specified using the same categories that describe the main method. Sub-method descriptions are embedded in method descriptions.

Capability Description: The PSM Process

Environment A characterisation of the operating environment of the KBS. Environmental conditions are a task feature in CommonKADS.

Resource Constraints Constraints on resources used during execution of the PSM. Costs are a similar task feature in CommonKADS.

Actor Constraints Constraints on agents involved in the performance of the method.

World Constraints: Data Input Constraints on data input to the PSM. Form, content and time pattern of inputs are task features in CommonKADS.

World Constraints: Data Output Constraints on data output by the PSM. Form, content and time pattern of outputs are task features in CommonKADS.

Sub-activities A specification of sub-methods of the PSM, defined in terms of the PSM Process categories.

The example in Figure 3 may help to clarify the index scheme. The method documented is cover and differentiate. Note that the example instances are shown as constants, sets, tuples, strings or boolean expressions (=True) where we think appropriate.[4]

[4]A proper ontology could be defined to represent the generic components of PSMs. Such an ontology could be included as a plug-in extension to SPAR.

```
---------------------------------------------------
Capability of Cover and Differentiate for Diagnosis

Competence
      Goal                  :diagnosis
      Problem Type          :{diagnosis}
      Generic Solution      :fault-cause
      Solution Component    :{case-model}
      Solution Property     :consistency(Complaint,
                             Diagnosis)=True
      Rationale             :''This method should be
         used when a causal theory of the
         behaviour of the system is available.
         The cover inference has input knowledge role
         Complaint and output knowledge role Hypothesis.
         [Complaint]-cover->[Hypothesis] ...''
Configuration
      Method Ontology        :causal-theory-of-behaviour
      Domain Theory
            Field            :{motor-vehicles}
            Ontology/Mapping :{(engineering-Cyc,equality)}
            Representation   :CycL
      Sub-methods            :{}
PSM Process Description
      Environment            :software-installed(Cyc)=True
      Resource Constraints   :cpu-cycles-free(99,%)=True
      Actor Constraints      :currently-executing(Cyc)=True
      World Constraints
       - Data Input    :{(Complaint,<set-of-concepts>)}
       - Data Output   :(Diagnosis,<concept>)
      Sub-activities         :none
---------------------------------------------------
```

Figure 3: An example capability description

Variants of the cover and differentiate method are discussed by Akkermans [2] who characterises the competence of several possible refinements in terms of consistency between complaints, observations and the diagnosis, or consistency between the complaint and the diagnosis. In this case, our description of the various refinements would differ in the Solution Property slot in our indexing scheme.

Other classes of diagnostic methods have been identified by Bredeweg [3] and these differ from the above regarding (at least) the Generic Solution, Solution Component and Method Ontology specifications, but share the same Goal and Problem Type attributes. For example, consistency-based diagnosis methods generate sets of faulty components as the Generic Solution, and find a solution, as opposed to an explanation, as the Solution Component. These methods require models of normal, abnormal, or both normal and abnormal behaviour, an attribute which is specified by Method Ontology slot.

F.6 Discussion

Our proposed set of categories may prove to be incomplete: We expect extensions and refinements from within the HPKB PSM working group, and from related research threads, including SPAR and KQML where the problems of specifying the objectives and capabilities of agents, plans and processes are also being considered. We believe that our approach highlights many issues which need to be considered when classifying PSMs, and that the organisational schemes we have used (SPAR process descriptions and the select/configure/execute classification) are well justified. We have begun to classify CommonKADS PSMs using this framework and have found the categories to make useful distinctions between methods.

The proposed set of categories may need to be extended to support functional requirements that we have not yet considered, formal verification being one example. The formality of the terms used to instantiate each category will also be dependent on the use to which the capability description is being put: Goals and Solution Properties can be formulated as axioms (as in competence-directed refinement [3]), and the inference structure can be formally represented [1]). This a requirement for the task of formal verification. Formal representations will also be required if selection or configuration processes are to be automated. We have chosen semi-formal representations as we are primarily concerned with identifying the most important categories, and have assumed the agents involved to be human.

We have not proposed an ontology for the values of each capability attribute. A widely-accepted ontology is needed in order to identify concepts as Goals or as Problem Types. Even in a more formal approach, where Goals are specified by axioms, the need for agreement about the use of symbols would remain. A proposal for such an ontology is made in [10] where a PSM description language is presented: PSMs and ontologies are described by frames where the slots can be filled by text, or by logical expressions in (an extension of) KIF. Only one competence attribute can be specified for a PSM, namely, constraints across inputs and outputs. This corresponds to a Solution Property in our proposal. However, a foundation ontology for commonly-used data structures, data types, and functions is proposed. *Constraints* and *fixes* are also classes in this ontology, and this is precisely the sort of ontology that is required to give meaning to the values of the attributes in the example capability description in Figure 3.

In conclusion, our approach has been to first identify the types of statements needed to characterise PSMs, prior to specifying an ontology of the problem domain, and an appropriate language. We have built upon existing work on problem solving methods to propose a general scheme for indexing PSMs which can be extended to account for new uses, and for the results of related research initiatives.

Acknowledgements

This work is sponsored by the Defense Advanced Research Projects Agency (DARPA) under grant number F30602-97-1-0203. The U.S. Government is authorised to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of DARPA,

Rome Laboratory or the U.S. Government.

References

[1] Aben, M. *Formal Methods in Knowledge Engineering* Ph.D. thesis, University of Amsterdam, 1995.

[2] Akkermans, H., Wielinga, B.J.,and Schreiber, G. Refinement: Competence Directed. in *Expertise model definition document.* (ed.) Wielinga, B.J. Technical Report ESPRIT Project P5248, KADS-II/M2/UvA/026.5.0, University of Amsterdam, 1994, pp. 117–135.

[3] Bredeweg, B. Model-based diagnosis and prediction of behaviour. in Breuker. J.A., Van de Velde, W. (eds.) *Expertise model document part II: The CommonKADS library.* pp. 113–148.

[4] Breuker. J.A., Van de Velde, W. (eds.) *Expertise model document part II: The CommonKADS library.* KADS-II/TM.2/VUB/TR/054/3.0 University of Brussels, 1994.

[5] Breuker. J.A. Problems in indexing problem solving methods. *Proceedings of the Problem-Solving Methods for Knowledge-Based Systems Workshop*, IJCAI 97, (ed). Fensel, D. pp. 19–35.

[6] Corby, O. and Dieng, R. Cokace: A Centaur-based environment for CommonKADS conceptual modelling language. *Proceedings of ECAI 96*, pp. 418–422. Also available at URL: `http://zenon.inria.fr/acacia/Cokace/cokace.html`

[7] Fensel, D. An ontology-based broker: Making problem-solving methods reuse work. *Proceedings of the Problem-Solving Methods for Knowledge-Based Systems Workshop*, IJCAI 97, (ed). Fensel, D. pp. 45–58.

[8] Filby, I. *Recommendations on standardisation of conceptual-level knowledge modelling formalisms.* Technical Report ESPRIT Project 9806 (EuroKnowledge), EuroK/T/010496-1/AIAI, November 1996.

[9] Gamma E., Helm R., Johnson R. and Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional Computing Series, 1995. ISBN 0-201-63361-2.

[10] Gennari, J.H., Grosso, W., and Musen, M. A method-description language: An initial ontology with examples. to appear in *Proceedings of KAW 98*, and available at URL: `http://ksi.cpsc.ucalgary.ca/KAW98S/KAW98S.html`

[11] Lydiard, T. *Using IDEF3 to capture the Air Campaign Planning process.* URL: `http://www.aiai.ed.ac.uk/~arpi/ACP-MODELS/ACP-OVERVIEW/96-MAR/HTML-DOC/idef3.html`

[12] Neumann, B. *Towards libraries of problem-solving models - candidates for standardisation.* Technical Report ESPRIT Project 9806 (EuroKnowledge), EuroK/T/960615-2/DTK, June 1996.

[13] Park, J.Y., Gennari, J.H., and Musen, M. Mappings for reuse in knowledge-based systems. to appear in *Proceedings of KAW 98*, and available at URL: `http://ksi.cpsc.ucalgary.ca/KAW98S/KAW98S.html`

[14] Schreiber, G., Wielinga, B.J., Akkermans, H., Van de Velde, W., and Anjewierden, A. CML: The CommonKADS conceptual modelling language. *Proceedings of the Eighth European Knowledge Acquisition Workshop*, LNAI 867, Steels, L., Schreiber, G., and Van de Velde, W. (eds.), Springer Verlag 1994, pp. 1–25.

[15] Shared Planning and Activity Representation (SPAR) URL: `http://www.aiai.ed.ac.uk/~arpi/spar`

[16] Steels, L. Components of expertise. *AI Magazine*, Summer 1990, pp. 28–49.

[17] Swartout, W., Gil, Y., Valente, A., and Tallis, M. *Knowledge acquisition for large knowledge bases: Integrating problem-solving methods and ontologies into applications.* URL: `http://www.isi.edu/expect/sherpa/sherpa.html`

[18] Tate, A. Roots of SPAR. to appear in the *Knowledge Engineering Review, Special Issue on Putting Ontologies to Use*, (eds.) Tate, A., and Uschold, M.F., 1998.

[19] Wielinga, B.J., Schreiber, T., and Breuker. J.A. KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition* 4, 1992, pp. 5–53.

G Integrating Problem-Solving Methods into Cyc

Stuart Aitken and Dimitrios Sklavakis

Proceedings of the 16th International Joint Conference on Artificial Intelligence, ed. Dean, T., Stockholm, 3-6th August 1999, Morgan Kaufmann, pp. 627–632.

Abstract

This paper argues that the reuse of domain knowledge must be complemented by the reuse of problem-solving methods. Problem-solving methods (PSMs) provide a means to structure search, and can provide tractable solutions to reasoning with a very large knowledge base. We show that PSMs can be used in a way which complements large-scale representation techniques, and optimisations such as those for taxonomic reasoning found in Cyc. Our approach illustrates the advantages of task-oriented knowledge modelling and we demonstrate that the resulting ontologies have both task-dependent and task-independent elements. Further, we show how the task ontology can be organised into conceptual levels to reflect knowledge typing principles.

G.1 Introduction

Developing reusable ontologies which specify the structure and content of domain knowledge has become a central problem in the construction of large and scalable knowledge based systems. For example, a key step in KBS construction using the Cyc system [7] is to extend the existing upper-level ontology by creating new classes and representations. Methodologies for ontology development have been proposed [7, 12, 1]. However, many unsolved problems remain. Other important issues concern the relationship between the domain representation and its intended use [14, 3]. We shall concentrate on the representational and performance issues focusing initially on the reasoning processes, and reflect on the implications for domain representation in the light of these findings.

Versions of Cyc are currently being used as an integration platform by the DARPA-funded High Performance Knowledge Bases (HPKB) program. Key issues on the HPKB program are the **scalability, robustness**, and **reusability** of knowledge-based system solutions. Cyc is unique in that it has potential solutions to each of these problems.

Cyc uses a resolution-based inference procedure that has a number of optimisations that improve the scalability of the architecture. For example, a specialised taxonomic reasoning module replaces the application of the logical rule for transitivity of class membership. Where specialised modules are not implemented, Cyc makes use of weak search methods to perform inference. Cyc lacks any principles for structuring inference at a conceptual level. Problem-solving methods provide precisely this structure, hence the importance of integrating structuring principles into a scalable KBS architecture.

Robustness and reusability are related properties of the knowledge representation scheme and the inference rules: Predicates such as *bordersOn* and *between*, defined in the upper-level ontology, can be reused in many different contexts. The combination of predicate properties (such as symmetry) and existing inference rules means that the use of these predicates is robust. Reconciling units of measure is a similar problem. In this case, Cyc has sufficient knowledge to prove *(greaterThan (Meter 1) (Centimeter 2))* using its existing definitions and rules about units of measure. Reusability is also an important motivation for defining a upper-level ontology as the basis of knowledge representation. The upper-level ontology can be shared among more specialised reasoning contexts or applications. Extensions to the upper-level can themselves be shared, and can be regarded as ontologies in their own right.

We describe the implementation of a PSM for fault diagnosis in Cyc. The diagnostic method was applied to two different domains to investigate whether the potential for method reuse was actually achievable. As implementation was preceded by a significant amount of domain and task analysis, this work allows us to review the value of the methodological approach and to investigate issues such as the task-dependence of the ontologies constructed. This paper begins with an introduction to the component technologies used—CommonKADS and Cyc—and then describes the implementation of the PSM and the associated knowledge modelling.

G.2 Component Technologies

G.2.1 PSMs: The CommonKADS View

In CommonKADS, problem-solving methods are the product of expertise analysis - one of several analysis steps which are specified by the methodology. PSMs are also used in Protege [10], and in Expect [2] (although in different forms). PSMs define distinct methods for performing a task, for example, diagnosis can be modelled as involving a heuristic association between observations and solutions, or as a process of decomposing a system and testing its subcomponents for correct operation. In addition to specifying an inference procedure, PSMs require that domain knowledge be modelled in particular ways, i.e. a method ontology is associated with a PSM.

CommonKADS is a methodology for KBS development which addresses not only the desired problem-solving performance of the end system, but the context in which it will operate. A number of models are constructed in the analysis phase: an *organisational model* represents the processes, structure, and resources of the organisation which is to use the KBS, a *task model* describes the activities of the process of interest, an *agent model* represents the agents involved in the process and their capabilities, a *communication model* describes agent (human and machine) communication, an *expertise model* defines domain and problem-solving knowledge, and, finally, a *design model* describes the structure and function of the system that will implement the knowledge-based task. More details of the various models, and appropriate modelling techniques can be found in [5].

CommonKADS is relatively neutral on questions of implementation. However, expertise modelling does make a number of assumptions about knowledge representation constructs and their interaction. The expertise model has three layers: the domain layer represents knowledge about the domain, the inference layer defines the procedures applied during problem solving, and the task layer specifies the ordering of inference steps. As the expertise model is the only CommonKADS model that captures expert problem-solving behaviour, we shall limit our attention to representing this model in Cyc.

G.2.2 Cyc

Cyc is a very large, multi-contextual knowledge-based system which is currently being used commercially by **Cycorp**. Cyc is also used for research purposes, and, in the HPKB program, Cyc is being used as a platform for technology integration.

The arguments for Cyc proposed in Lenat and Guha [1990] remain the cornerstones of the Cyc project; namely, the need to overcome the brittleness of traditional expert systems, and the means of achieving this through the development of a shared ontology representing 'consensus reality'. The upper-level ontology, which constitutes the basis of knowledge representation in Cyc, has been made publicly available. However, this represents only a fraction of the knowledge which has been entered into Cyc.

The upper-level ontology is represented in a variant of first-order logic known as *CycL*. The ontology includes: classes used for constructing representations, for example *SetOrCollection* and *Predicate*; classes for high-level concepts such as *Event* and *Agent*; and more specific classes representing commonly occurring objects and events such as *Book* and *BirthEvent*.

Assertions in CycL are always associated with a *microtheory* context. The *BaseKB* contains the upper-level ontology and new contexts can be defined which specialise this theory. Multiple inheritance of microtheory contexts is allowed. Alternative specialisations of a microtheory need not be consistent with each other: a microtheory can contain ontology extensions and assertions which are inconsistent with those defined in a different theory - providing neither context is defined as subsuming the other. The microtheory mechanism plays an important role in structuring inference.

Cyc performs inferencing in response to a query by the user (by backward chaining) or in response to an assertion (by forward chaining with rules which are explicitly specified to be forward rules). Queries are made in a specific microtheory which forms the local search context. Typically, a microtheory will be a specialisation of one or more theories and in this case search will progress out to wider contexts should a solution not be found locally. Queries are treated in a purely logical manner: the order of conjuncts is not considered to be significant and may be changed by optimisations operating at the clause-form level. The preconditions of rules are also treated in this way - prohibiting the user from influencing the search process in a predictable way. The dependencies between derived facts, rules and assertions are recorded and maintained by a truth maintenance mechanism.

Cyc's purely declarative treatment of rules differs from other approaches to logic-based knowledge representation, such as Prolog, where the ordering of clauses, and of literals within clauses, is used to determine the order of search.

The Cyc system includes a number of tools for viewing and browsing the ontology. In common with other browsers, including that for Loom [9], terms in the ontology are hyperlinked in a web-based interface. This allows the user to explore the concepts which define, or are subsidiary to, the concepts currently being displayed.

The Cyc system also gives the KBS developer access to a LISP-like environment where new procedures can be defined in the *SubL* language. The Cyc knowledge base and inference engine can be accessed via the SubL functions *ask* and *assert*. Due to the treatment of rules described above, imposing structure on the search process necessarily requires SubL coding.

G.3 Systematic Diagnosis in Cyc

This section describes the expertise modelling process and presents its products. The implementation of these models in Cyc is then outlined. We begin with a brief introduction to the domain and the diagnostic task.

The task of diagnosis was selected because a set of well understood methods for solving such tasks already exists [13]. An important part of expertise modelling is the selection between alternative methods - with their accompanying behaviours and assumptions. The choice of a specific diagnostic method was not made prior to domain analysis. It is readily apparent that we have chosen a problem type that falls within the scope of the methodology we intend to apply. However, it is not at all obvious that diagnosis—which is inherently an incremental procedure requiring information gathering—can be adequately implemented by backward chaining driven by a query-based interaction (i.e. by the default environment provided by Cyc). We shall return to this point below.

Fault finding in personal computers (PCs) was chosen as the primary task domain. This task can be modelled accurately, i.e. the actual behaviour of human experts is known and has been documented [6], yet the amount of electronics knowledge required is low as fault finding never progresses to a level where sophisticated test equipment is required. The second domain chosen was fault finding in an automobile ignition system. This task ought to be soluble by the method developed for PC diagnosis, despite differences in the characteristics of the domain and in the method ontology.

G.3.1 Modelling Expertise

The selection of a problem-solving method is one of the central modelling decisions in CommonKADS. This will typically have an impact on domain representation. Following this approach, the PC-diagnosis problem was addressed by investigating candidate PSMs. As PSMs may be refined in several different ways, alternative instantiations were also investigated. This is a notable contrast with a domain-oriented approach which would focus on developing an ontology of the domain being reasoned about, PC systems and their components in this case.

The systematic diagnosis PSM was found to match the expert reasoning process most closely. The generic model had to be adapted to reflect expert reasoning more faithfully. The central steps in systematic diagnosis are the decomposition of the system being diagnosed into subsystems, and the testing of the subsystems for correct operation by making tests and comparing the observed with the predicted outcomes. The subsystem currently being tested is said to play the role of the current *hypothesis*. Testing may rule out this hypothesis, in which case another subsystem becomes the hypothesis. Testing may yield an inconclusive result, in which case more tests are required, or testing may indicate a fault, in which case the diagnosis is

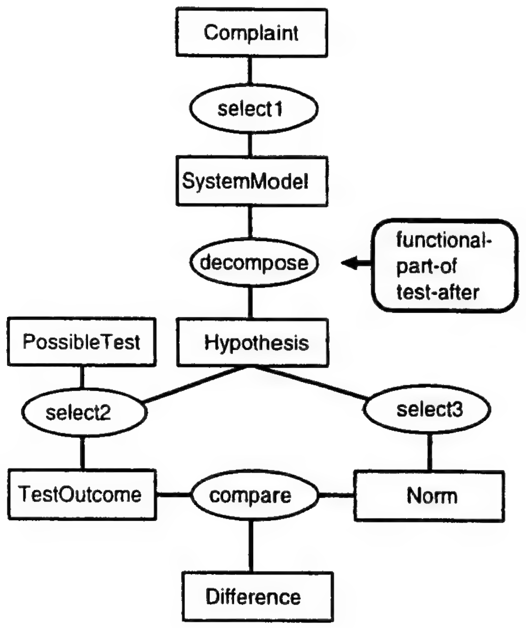


Figure 4: PSM for systematic diagnosis

concluded—if the current hypothesis cannot be further decomposed (i.e. it is a component), or diagnosis continues at a lower level of system decomposition—if the current hypothesis can be decomposed (i.e. it is a system). The system model may describe how the system is decomposed into (physical) parts, or may describe the functional relationships between systems.

It was discovered that the the *part-of* model, which lies at the heart of systematic diagnosis, had to be instantiated to *functional-part-of* in the PC diagnosis domain. That is, problem solving requires a functional view of the system, rather than a component/subcomponent view. The *functional-part-of* predicate is clearly a representational construct at the domain level, and is one of several part-of views that might be taken of a system. In fact, there was no need to represent the *physical-part-of* relation in order to solve this problem.

Another important refinement of the generic model was the addition of theories of test ordering. Where there are several decompositions of a system, the model permits any subsystem to play the role of hypothesis. However, in PC diagnosis it is important to establish first, for example, that the power system is operational, then that the video system is operational. Once the video system is known to work we can be sure that the results of BIOS system tests are being displayed correctly. Similar ordering constraints were found for all subsystems, and at all levels of decomposition. Consequently, there is a need to impose an order on hypothesis selection (or, equivalently, system decomposition) and we chose to represent this knowledge in a heuristic fashion via a *testAfter* predicate. Figure 4 shows the specialised PSM in a diagrammatic form.

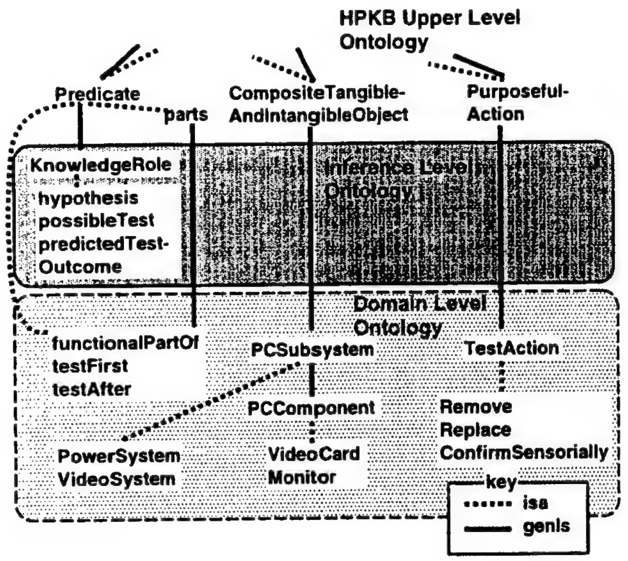


Figure 5: Upper-Level ontology extensions - distinguished by level

Determining the overall view of the desired problem-solving behaviour aided knowledge acquisition, much of which concerned the extraction and structuring of information from an on-line source [6]. Our experience confirmed the claimed advantages of the modelling approach. In addition to specifying an inference-level procedure, knowledge acquisition also requires the content and scope of domain knowledge to be determined. The task of representing domain knowledge in Cyc followed the standard procedure of extending the ontology by defining new collections and predicates, and linking these to existing constants. We now describe the Cyc implementation in more detail.

G.3.2 Cyc Implementation

Diagnosis requires interactive data gathering, and the subsequent evaluation of test results and updating of the current hypothesis. Such a procedure cannot be implemented by logical inference alone, and so it is clearly necessary to use Cyc's LISP-like language, SubL, to implement a control regime. In CommonKADS, control knowledge is divided between the inference layer, where knowledge roles and inference steps are defined, and the task layer, where the order of application of inference steps is specified. Our aim was to represent the levels of the expertise model in Cyc in as faithful a manner as possible. We begin by considering domain knowledge.

Domain knowledge was represented by extending existing collections where possible. Figure 5 shows a small illustrative set of the extensions made. The collection PCSubsystem was added as a subcollection of CompositeTangibleAndIntangibleThing, and PCComponent was defined as a specialisation of it. Both types of object have a tangible component, and may carry information hence have an intangible component also. TestAction was defined as a new collection of PurposefulAction, and the instances of Remove, Replace, and ConfirmSensorially (i.e. confirm by observation) were added [11]. functionalPartOf was introduced to represent the functional decomposition of a system, and stated to generalise to parts, being the most general existing *part-of* predicate in the upper ontology[5]. Other specialisations of parts include physicalDecompositions and timeSlices.

The predicates testFirst and testAfter were introduced as predicates to represent the test ordering theory. A test is defined by three components: a TestAction, a PCSubsystem and a PossibleObservable. The collections PossibleObservable, PossibleObservableValue, and ResultType were defined as subcollections of AttributeValue. The representation of testing knowledge can be made more robust by grounding it extensively in the upper ontology. In contrast, part-of facts are not likely to be derivable by appeal to background knowledge.

At the inference level, knowledge roles are represented by predicates, and inference steps are rules which have knowledge roles as preconditions and conclusions. Figure 5 shows the introduction of the KnowledgeRole collection, a specialisation of the Predicate class of the upper ontology. Instances of KnowledgeRole predicates take domain-level formulae or collections as arguments. Examples include; the unary predicate hypothesis - applicable to PCSubsystem - denotes the current hypothesis, possibleTest holds of applicable tests, and the relation predictedTestOutcome holds of a test, PossibleObservableValue and a ResultType. More complex mappings to the inference level, and the definition of additional collections and terms, are also possible within this approach. The CycL language is sufficiently expressive to allow complex mappings of the type described in [14] where the inference-level ontology (in our terminology) might define relations holding of domain-level ontology, e.g. we could express the fact that PhysicalPartOf is a relation: *relation(PhysicalPartOf)*.

In a similar way, the currently invoked inference step (e.g. *decompose*, *select*) is also explicitly asserted in the KB by predicates which belong to the inference level.

Inference steps are invoked by querying or asserting knowledge roles. For example, the role *hypothesis* holds of the subsystem currently playing the role of the hypothesised fault. The rules for selecting the test ordering theory depend on the current hypothesis, for example:

```
F: (implies (and (hypothesis PCSystem)
                 (plausibleInference Decompose))
       (and (testFirst PowerSystem)
            (testAfter PowerSystem VideoSystem))).
```

This is a forward rule which fires when *hypothesis* and *plausibleInference* are asserted. The current hypothesis assertion must be deleted and replaced as diagnosis proceeds. These operations are implemented in SubL by functional-interface functions, within the larger structure of the systematic diagnosis task function. The user could make this series of deductions themselves, and in the implemented system, the user is able to inspect the state of the reasoning process as it progresses. As an example, the following SubL code is called at the start of diagnosis and

[5]Note that terms defined in the ontology, or its extension, are written in sans-serif, following the Cyc convention, names of collections begin with a capital letter and predicates begin with a lower case letter.

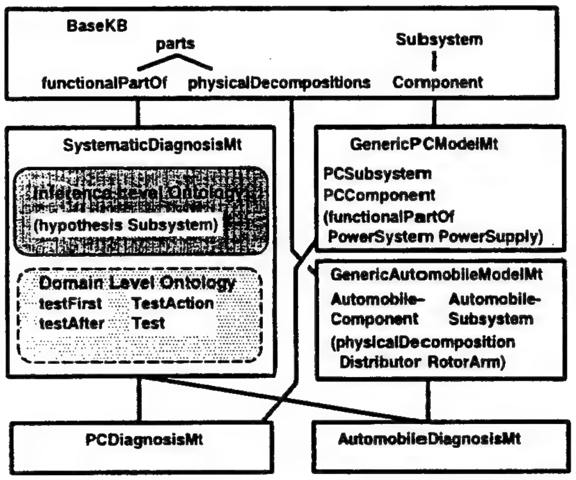


Figure 6: Microtheory structure

simply asserts that the entire system is the hypothesis, and then calls another SubL function, `sd-select2-3`, which performs system decomposition.

```
(define sd-select1 (system)
 (fi-assert (#$hypothesis system) *defaultMt*)
 (sd-select2-3))
```

We have achieved an explicit representation of knowledge roles and of inference steps in Cyc that reflects the knowledge typing principles advocated in [3]. Control over the search process is achieved by making a series of simple queries, structured to implement the pattern of inference of the PSM. We found no need to extend the functionality of Cyc, or the expressivity of its representation language(s) in order to implement the PSM. The central problem was to combine the available features into structured architecture, in order to to take advantage of the model-based approach to problem solving.

We tested the reusability of the domain and inference level definitions, and of the SubL code, by considering diagnosis in the domain of automobile ignition systems. This experience is discussed in the wider context of the reusability, scalability, and robustness of our approach.

G.4 Representation and Reasoning

G.4.1 Domain Ontologies

The view of domain ontology construction which results from the prior selection and adaption of an explicit problem solving method is more focussed on concepts relevant to problem solving than a task-neutral view would be. The resulting domain ontology is not task-specific in its formalisation, e.g. the definition of the *functional-part-of* relation has no intrinsic task-related properties. But, the coverage of the resulting ontology may only be partial – we did not need to elicit *physical-part-of* knowledge.

Had we taken a view that focussed on the domain alone, we would have had no explicit guidance as to which concepts were or were not relevant to the ontology definition effort. We have gained experience of constructing ontologies where the primary aim was to represent the domain, with ontology definition only informally guided by considering the task. Under these conditions it is difficult to determine the relevance of a potential domain concept, and the distinction between concepts that are intrinsic to the representation of the domain, and those that are related to the task to be performed was difficult to make.

Reusability of domain knowledge is an important issue, and our approach has been to use the microtheory mechanism of Cyc to encapsulate the generic components of the extended ontology. The resulting mircotheory structure, shown in Figure 6, places the generic system models for PCs and automobile systems in distinct microtheories, that are extensions of the BaseKB, and are included in the specific diagnosis microtheories. Strictly speaking, these microtheories are not extensions of the ontology as they make no new specifications. Instead, the BaseKB is extended by adding the definitions of the functionalPartOf predicate and the collections Subsystem and Component as these concepts are sufficiently general to be reusable across domains. The method-specific ontology, comprising domain and inference level components, is also a specialisation of the BaseKB, and this theory is shared by both PC and Automobile diagnosis theories. The microtheory structure shows that the generic system models can be used in any context which includes the (now extended) BaseKB, and that these theories can be thought of as parameters of the diagnosis microtheories.

G.4.2 Inference Knowledge

The application of systematic diagnosis to the automobile domain required a change in system theory from *functionalPartOf* to *physicalDecomposition*. While this is a significant change in the modelling of the diagnostic process (physical parts play the role of hypotheses) there were few implications for formalisation of the inference level as no new knowledge roles were found. Similarly, the SubL code was only modified to take the specific diagnosis microtheory as a parameter. In future, we aim implement other PSMs and this may permit us to generalise inference-level theories across PSMs.

G.4.3 Scalability

The domain and inference level knowledge representations that we have used are extensions of the basic representation, and can make use of the existing optimisations for indexing large KBs, performing taxonomic reasoning and theory structuring. Our approach to PSM implementation is based on structuring a series of queries and assertions to implement a problem-solving method. As the individual queries are simple, the space searched is small (we can specify the depth of search to be 1–3 levels). This contrasts with the basic query mechanism where the only means of getting an answer to query which requires many rules to be combined is to increase the depth of search - with the resulting exponential increase in search space.

G.4.4 Robustness

At present we are unable to reason about inference structures or about the mappings from the domain to the inference level within Cyc. There are no rules which allow PSMs to be modified or to be configured. Consequently, the system lacks robustness as it cannot fall back to first principles when an existing method is not immediately applicable. The problems of PSM modification and configuration are significant, even for human experts, but we believe that automatically specialising PSMs is a feasible proposition. We also plan to explore the idea of falling back to more general methods, when more specific methods are inapplicable, to regain robustness.

Inference steps (implemented by rules) require proving domain-level predicates, and robustness at the level of reasoning about domain knowledge occurs exactly as in Cyc.

G.5 Discussion

We have described an approach to implementing problem-solving methods in Cyc which makes use of the existing optimisations developed for large-scale knowledge bases, and adds additional structure to the inference process. Extensions to the existing ontology distinguished generic extensions to the upper-level ontology, extensions to the knowledge base, and task-related extensions. Knowledge typing principles were used within the task-related ontology to further structure problem-solving knowledge.

Our investigation of diagnostic problem solving has not only raised issues of knowledge reuse and scalability, but also of system-environment interaction. Intelligent systems cannot rely on large amounts of background knowledge alone as many classes of problems require information gathering or user interaction. If such interaction is to happen in an intelligent fashion then there is a requirement to represent and reason about the inferences which require interaction.

Acknowledgments

This work is sponsored by the Defense Advanced Research Projects Agency (DARPA) under grant number F30602-97-1-0203. The U.S. Government is authorised to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of DARPA, Rome Laboratory or the U.S. Government.

References

[1] Blázquez, M., Fernández, M., Garcia-Pinar, J.M., and Gómez-Pérez, A. Building Ontologies at the Knowledge Level using the Ontology Design Environment. *Proceedings of KAW'98* Banff, 1998.
URL: `http://ksi.cpsc.ucalgary.ca/KAW/KAW98/`
`blazquez/`

[2] Gil, Y. and Melz, E. Explicit representations of problem-solving strategies to support knowledge acquisition. *Proceedings of the Thirteen National Conference on Artificial Intelligence (AAAI-96)*, Portland, OR, August 4-8, 1996.

[3] van Heijst, G., Schreiber, A.T., and Wielinga, B.J. Using Explicit Ontologies in KBS Development. *International Journal of Human-Computer Studies*, Vol. 46, No. 2/3, pp. 183–292.

[4] HPKB Information about the HPKB program can be found at URL: `http://www.teknowledge.com/HPKB/`

[5] Kingston, J., Griffith, A., and Lydiard, T. Multi-Perspective Modelling of the Air Campaign Planning Process. *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, Nagoya, August 23-29 1997, pp. 668–673.

[6] Kozierok, C.M. *Troubleshooting Expert* PCGuide URL: `http://www.pcguide.com`

[7] Lenat, D.B. and Guha, R.V. *Building large knowledge-based systems. Representation and inference in the Cyc project.* Addison-Wesley, Reading, Massachusetts, 1990.

[8] Lenat, D.B. *Leveraging Cyc for HPKB Intermediate-level Knowledge and Efficient Reasoning*
URL: `http://www.cyc.com/hpkb/`
`proposal-summary-hpkb.html`

[9] *The Loom tutorial* Artificial Intelligence research group, Information Sciences Institute, USC, December 1995.
URL: `http://www.isi.edu/isd/LOOM/`
`documentation/tutorial2.1.html`

[10] Puerta, A.R., Egar, J.W., Tu, S.W., Musen, M.A. A multiple-method knowledge-acquisition shell for the automatic generation of knowledge-acquisition tools. *Knowledge Acquisition* 4, 1992, pp. 171–196.

[11] Sklavakis, D. *Implementing Problem Solving Methods in Cyc.* M.Sc. Dissertation, Division of Informatics, University of Edinburgh, 1998.

[12] Uschold, M. and Gruninger, M. Ontologies: principles, methods and applications. *Knowledge Engineering Review*, Vol. 11:2, 1996, pp. 93–136.

[13] Wielinga, B.J., Schreiber, T., and Breuker. J.A. KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition* 4, 1992, pp. 5–53.

[14] Wielinga, B.J., Schreiber, G., Jansweijer, W., Anjewierden, A., and van Harmelen, F. Framework and formalism for expressing ontologies. KACTUS Project Report (Esprit Project 8145) DO1b.1-Framework-1.1-UvA-BW+GS+WJ+AA, University Of Amsterdam, 1994.

Knowledge representation and Reasoning: Challenge problems

This section summarises the work that AIAI did to contribute to solving (as opposed to acquiring knowledge to help others solve) two of the Challenge Problems: the Year 1 Workarounds Challenge problem and the year 2 Course of Action critiquing Challenge problem.

H High Performance Knowledge Bases: Four approaches to Knowledge Acquisition, Representation and Reasoning for Workarounds Planning

Publication details: Submitted to "Expert Systems with Applications", Elsevier Science, for publication in 2000. THis document constitutes deliverable P-5.

John Kingston

Purpose: To compare and contrast the four approaches used to solve the Workarounds planning challenge problem.

Abstract: As part of the DARPA-sponsored High Performance Knowledge Bases program, four organisations were set the challenge of solving a selection of knowledge-based planning problems in a particular domain, and then modifying their systems quickly to solve further problems in the same domain. The aim of the exercise was to test the claim that, with the latest AI technology, large knowledge bases can be built quickly and efficiently. The domain chosen was 'workarounds'; that is, planning how a convoy of military vehicles can "work around" (i.e. circumvent or overcome) obstacles in their path, such as blown bridges or minefields.

This paper describes the four approaches that were applied to solve this problem. These approaches differed in their approach to knowledge acquisition, in their ontology, and in their reasoning. All four approaches are described and compared against each other. The paper concludes by reporting the results of an evaluation that was carried out by the HPKB program to determine the capability of each of these approaches.

High Performance Knowledge Bases: Four approaches to Knowledge Acquisition, Representation and Reasoning for Workaround Planning

John Kingston
AIAI, Division of Informatics, University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN
J.Kingston@ed.ac.uk
Tel. +131 650 2732 FAX +131 650 6513

Acknowledgements

The work described in this paper has been contributed to by many people: Gheorghe Tecuci, Mihai Boicu, Katie Wright and Mike Bowman at the Learning Agents Laboratory of George Mason University; Yolanda Gil, Bill Swartout and Jim Blythe at the Information Sciences Institute, University of Southern California; Ben Rode, Keith Goolsbey, Fritz Lehmann and Doug Lenat at Cycorp; Adam Pease and Cleo Condoravdi at Teknowledge; Eric Jones and Eric Domeshek at Alphatech; and Stuart Aitken at AIAI. Mention should also be made of David White, the primary domain expert; Dave Gunning and Murray Burke, who have managed the HPKB program; and Albert Lin at SAIC.

Running title: HPKB: Four approaches to Workaround Planning

High Performance Knowledge Bases: Four approaches to Knowledge Acquisition, Representation and Reasoning for Workaround Planning

Abstract

As part of the DARPA-sponsored High Performance Knowledge Bases program, four organisations were set the challenge of solving a selection of knowledge-based planning problems in a particular domain, and then modifying their systems quickly to solve further problems in the same domain. The aim of the exercise was to test the claim that, with the latest AI technology, large knowledge bases can be built quickly and efficiently. The domain chosen was 'workarounds'; that is, planning how a convoy of military vehicles can "work around" (i.e. circumvent or overcome) obstacles in their path, such as blown bridges or minefields.

This paper describes the four approaches that were applied to solve this problem. These approaches differed in their approach to knowledge acquisition, in their ontology, and in their reasoning. All four approaches are described and compared against each other. The paper concludes by reporting the results of an evaluation that was carried out by the HPKB program to determine the capability of each of these approaches.

Introduction

The goal of the DARPA-sponsored High-Performance Knowledge Base (HPKB) program, which ran from 1997 to 1999, was to produce the technology needed to enable system developers to construct rapidly large knowledge-bases (with many thousands of axioms) that provide comprehensive coverage of topics of interest, are reusable by multiple applications with diverse problem-solving strategies, and are maintainable in rapidly changing environments. In the original proposal, it was envisioned that the process for constructing these large, comprehensive, reusable, and maintainable knowledge bases would involve three major steps:

- Building Foundation Knowledge: creating the foundation knowledge (e.g., selecting the knowledge representation scheme, assembling theories of common knowledge, defining domain-specific terms and concepts) to enable the construction and population of large, comprehensive knowledge bases for particular domains of interest -- by selecting, composing, extending, specializing, and modifying components from a library of reusable ontologies, common domain theories, and generic problem-solving strategies.
- Acquiring Domain Knowledge: constructing and populating a complete knowledge base -- by using the foundation knowledge to generate domain-specific knowledge acquisition, data mining, and information extraction tools -- to enable collaborating teams of domain (non-computer) experts to easily extend the foundation theories, define additional domain theories and problem solving strategies, and acquire domain facts to populate a comprehensive knowledge base covering the domains of interest.
- Efficient Problem Solving: enabling efficient problem solving -- either by providing efficient inference and reasoning procedures to operate on a complete knowledge base, or by providing tools and techniques to select and transform knowledge from a complete knowledge base into optimized problem-solving modules tailored to the unique requirements of an application.

The objective of HPKB was to develop, integrate, and test the technology needed to enable this process. The intention was to produce alternative knowledge-base development environments, which combined the necessary foundation-building, knowledge-acquisition, and problem-solving technologies into an integrated development environment, and to use those environments to build reusable knowledge-base components for multiple DARPA application projects.

Challenge Problems

In the first year of the HPKB project, progress towards these goals was encouraged by setting up three competitive scenarios in which several technology developers were tasked to tackle a knowledge-based problem. These were known as "challenge problems". Each problem consisted of a collection of data and a set of sample problems with model answers; an evaluation was performed at the end of the period of development, in which the systems were tested on their ability firstly to handle new problems based on the same knowledge, and secondly to add new knowledge (in the same domain) rapidly, and to answer questions that drew on the new knowledge.

The common availability of input data in agreed formats, and the co-ordination of multiple technology developers in tackling a single challenge problem, was handled by two companies (Teknowledge and SAIC) acting as integrators; the efforts of the technology developers were thus co-ordinated into two "teams". The two teams took slightly different approaches, which are reflected in some of the systems developed; Teknowledge favoured a centralised architecture based on a large common-sense ontology (Cyc) while SAIC had a distributed architecture that relied on sharing specialised domain ontologies and knowledge bases, including a large upper-level ontology based on the merging of Cyc, SENSUS (Swartout et al, 1996) and other knowledge bases.

The three Challenge Problem scenarios that were set up in the first year of the HPKB project were:

- Crisis Management. This work linked up with another DARPA project (GENOA) which aimed to help intelligence analysts understand emerging international crises more rapidly. A scenario was developed in which hostilities between Saudi Arabia and Iran lead to the closure of the Strait of Hormuz (at the mouth of the Persian Gulf) to international shipping. Technology developers were then given the task of building systems to answer situation assessment questions, such as "Is Iran capable of firing upon tankers in the Strait of Hormuz?" and "What risks would Iran face in closing the Strait?" These are questions about motives, intents, risks, rewards and ramifications that may have multiple answers, and so significant common-sense reasoning is required to determine the most plausible answers to these questions.
- Movement Analysis. Given an idealised dynamic radar image showing vehicles moving in an area of several thousand square miles, the challenge problem was to identify types of vehicles (e.g. slower-moving dots travelling in convoys may be military vehicles) and to identify strategic military locations (e.g. places visited regularly by military vehicles).
- Workarounds. If a road or track is blocked by a large object, a crater, a minefield, or a blown bridge, there are a number of ways of "working around" that obstacle. The challenge problem was to calculate the swiftest way of working around an obstacle, given data about the nature of the obstacle, the terrain, and the availability and location of specialised assets such as portable bridges.

Each of these challenge problems required different AI technology to solve it. The Crisis Management scenario required text understanding and a detailed ontology and knowledge base in order to support something close to common sense reasoning. Movement Analysis required spatial reasoning and fusion of multiple inputs. Workarounds planning required knowledge-based planning. For more details of each of these challenge problems, see the comprehensive paper on the HPKB project in AI Magazine (Cohen et al, 1998).

This paper focuses on the Workarounds challenge problem. Four technology developers provided solutions to this challenge problem: AIAI, Cycorp (with assistance from Teknowledge), George Mason University, and ISI (the Information Sciences Institute of the University of Southern California). This paper will describe the Workarounds planning challenge problem in more detail, describe each of the solutions in terms of their approaches to knowledge acquisition, ontology, and reasoning; and then present the results of the challenge problem evaluations as part of a critical evaluation of each approach.

The Workaround Planning Challenge Problem

The Workarounds challenge problem required deciding how to circumvent or overcome obstacles to military traffic. Through knowledge acquisition performed in the course of the first year by AIAI and others, it became clear that there were six different ways of circumventing or overcoming obstacles:

- Bridging gaps;
- Filling gaps;
- Reducing obstacles until they are trafficable (or demolishing them completely);
- Finding alternate routes;
- Providing alternate transport (e.g. replacing a bridge over a river with a ferry);
- Clearing minefields.

Each of these classes of solution had several instances; for example, bridging a gap can be done with an AVLB (a light bridge carried on an armoured vehicle), a medium girder bridge, a Bailey bridge, or a ribbon bridge. Each solution instance has its own constraints; for example, AVLBs require both banks to be fairly level, and have a maximum usable length of about 20 metres, while ribbon bridges can only be used on water.

The planning requirements of the problem become clear when it is realised that each solution instance may require multiple steps (e.g. first transport the AVLB to the gap site, then set it up); the various constraints on solutions may require further steps (e.g. one bank must be bulldozed to make it sufficiently level before an AVLB can be set up, which requires getting a bulldozer to the site); and a full workarounds solution may make use of more than one solution instance (for example, the approach to a blown bridge may be mined, requiring both mine clearance and bridging; or a river in a valley may be crossed by bulldozing the banks on both sides to create two alternate routes to the river's edge, and then bridging the river with a ribbon bridge). Workarounds plans usually require less than 20 plan steps, so full-scale AI planning systems are not essential, but some planning capability is needed to solve this problem satisfactorily.

The technology developers were provided with information on the transportation link, the obstacle to be worked around, and key features of the local terrain; the units (tanks or trucks) that would be likely to use that transport route; and a detailed description of resources (such as Army engineering units) in the area that could be used to repair the damage. They were also provided with the written and diagrammatic results of knowledge acquisition sessions conducted over the course of the year. The expected outputs were a reconstitution schedule (an estimate of the capacity of the damaged link over time, which requires a workaround plan), a time line of engineering actions needed to repair the link (if no alternate transport or alternate route is available, this will be the same as the workaround plan), and a set of assets required to effect the repair. From the point of view of the technology developers, this required considering alternate plans for repairing the link, calculating which plans were the most time-effective, and presenting the full details of these plans as outputs.

AIAI's approach: Hierarchical Task Network planning within CYC

AIAI approached the workarounds planning problem as a part of the Teknowledge integration team, and with a background in developing the O-Plan AI planner (Tate et al, 1996), and in extracting the reasoning "primitives" from O-Plan in order to allow declarative planning within a standard knowledge-based systems 'shell' (Kingston et al, 1996). These reasoning primitives consist of permitted activities in the domain, represented as task formalisms (TFs); each TF states the preconditions of an activity, the effects of that activity, and (if applicable) the sub-activities of that activity. In the Search and Rescue system (Kingston et al 1996; Cottam et al, 1995), the TFs were represented as CLIPS objects, and CLIPS rules were used to determine which objects could be inserted into the plan given the current plan state, as well as identifying activities that could be done in order to achieve a plan state needed by another key activity.

In many planning tasks, the various activities are all sub-activities of a single top level activity that needs to be achieved; in this case, the system is said to be performing hierarchical task network (HTN) planning. Many well-known planning systems have used HTN planning, including Nonlin, SIPE, and O-Plan. For workarounds planning, the overall goal is to get to the other side of the obstacle, so this can be set up as the top level activity; the six solution classes described above then become the six possible sub-activities of that top level activity, and the various solution instances, and steps that comprise those solution instances, form sub-activities at various lower levels of decomposition.

AIAI offered to build a "proof of concept" workarounds planner in Cyc, using HTN planning. The aim was to represent TFs in Cyc, and then to use Cyc's default reasoning module (which uses backward chaining on axioms that represent implications to determine whether a query can be proved) to construct a full plan. By working entirely within Cyc's capabilities for common-sense reasoning, AIAI hoped to make the system robust to real-world changes and modifications; an example of such a change would be that the problem of crossing a river or a lake is greatly reduced if the air temperature is significantly below 0 degrees Celsius. In the event, few such issues arose in the challenge problems.

AIAI's work was also intended to define a usable, general purpose ontology of planning within Cyc, and to show how reasoning could be performed on constants conforming to that ontology.
One of the key original aims of Cyc was to develop a system that is capable of common-sense reasoning; its developers quickly discovered that ontological accuracy is an essential prerequisite of accurate common-sense reasoning. Cycorp has therefore developed an ontological approach which is divided into three levels: the upper level (where generic predicates such as GeographicalRegion and TransportationDevice are defined), the lower level (where domain-specific or problem-specific predicates are defined), and an intermediate level. In addition, constants in Cyc are differentiated on dimensions such as "stuff-like" versus "object-like" (based on whether identity is retained when the thing is divided up; so water is stuff-like, whereas a human being is object-like), and "always true" versus "sometimes true" (or, more generally, what the *persistence distribution* of a constant is). The effects of these dimensions are mitigated or

magnified by context; readers interested in these ramifications are referred to (Lenat98).

From the viewpoint of the workarounds challenge problem, it became clear that in order to represent TFs and plans in Cyc, a sub-ontology of planning terms needed to be introduced into Cyc. This ontology is shown in Figure 1.

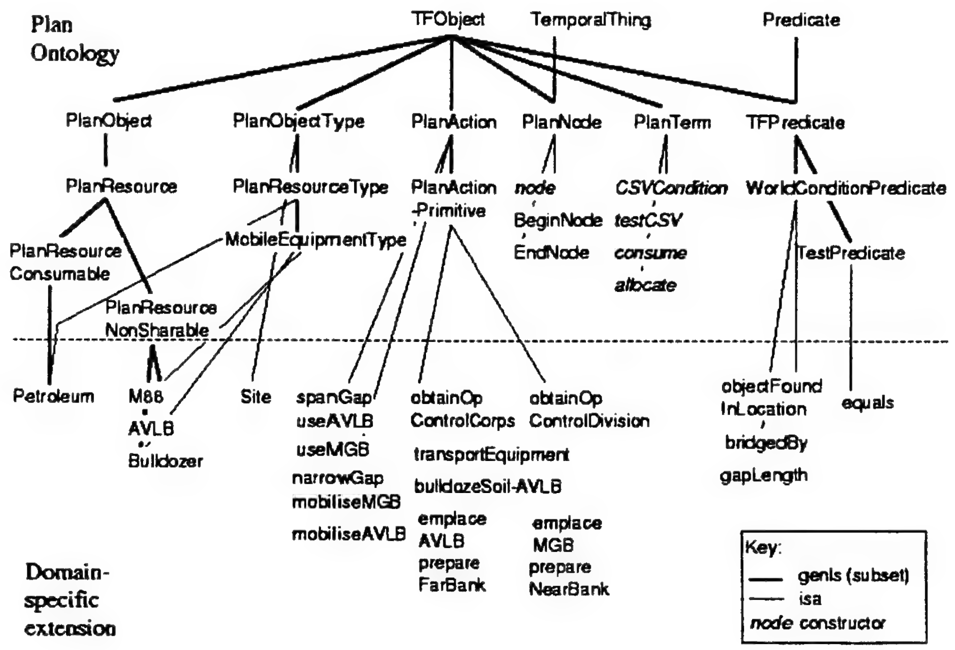


Figure 1: Plan ontology in Cyc (from (Aitken & Kingston, 1999))

These ontology definitions were used to create constants in Cyc that represented the TFs, the plan, the conditions of TFs, and the plan resources. For example, the following implication axiom (or 'rule') was created to test if a gap could be spanned by an AVLB:

```
(implies (isa ?AVLB AVLB)
 (potentialAction spanWithAVLB
  (conj (CSVCondition Site ?Site gapLength
           (testCSV lessThan (Meter 17.37)))
  (conj (CSVCondition Site ?Site riverBankMaxSlope
           (testCSV lessThan (Degree-UnitOfAngularMeasure 13.5)))
   (CSVCondition AVLB ?AVLB locationOf (testCSV equals ?Site))))
 (CSVCondition Site ?Site bridgedBy (testCSV equals ?AVLB)) ?I ?J)).
```

This rule states that if there is a site less than 17.37 metres wide, with banks sloped at no more than 13.5 degrees, and the AVLB is present at the site (according to the current plan state, represented by ?I), then a new node can be added to the current plan; at this node, the plan state (represented by ?J) considers the site to be successfully bridged by an AVLB.

As already indicated, the reasoning performed by AIAI's system was based on backward chaining through a set of these rules in order to generate a plan; in effect, Cyc is being 'tricked' into generating a plan when it thinks it is proving a goal. The version of Cyc which was used for this challenge problem (version 1.1) was able to generate plans using this method, but was not able to calculate the total time required for a plan; this calculation was done manually. More recent versions of Cyc have introduced this capability.

Knowledge acquisition for AIAI's planner was done by inspecting the published documents and data, and transforming this information into suitable rules and constants in Cyc. No supporting knowledge acquisition tool was used.

TFS/Cycorp's approach: Re-using pre-defined ontology in a Lisp-based planner

Cycorp were also part of the Teknowledge integration team; indeed, they worked sufficiently closely with Teknowledge that the integration work was done by Teknowledge with significant input from Cycorp, while the challenge problems were tackled by Cycorp with significant resource from Teknowledge. The technology developers were therefore referred to as the TFS team.

Like AIAI, TFS used no tool that acquired knowledge directly from experts. They did, however, create a translator that automatically generated information about a workaround problem, both from the specific inputs supplied by the developers of the challenge problem, and from other sources. This process of creating axioms based on other sources is (informally) known as "knowledge slurping". A selection of the axioms generated by this translator is shown in Figure 2.

```
(widthOfObject River6 (Meter 16))
(lengthOfObject Bridge1 (Meter 33))
(spans-Bridgelike Bridge1 Crevice3)
(in-ContOpen River6 Crevice3)
(bordersOn Bridge1 Approach1)
(bordersOn Bridge1 Approach2)
(gapWithinPath Bridge1 Damage1)
(isa Damage1 GapInPathArtifact)
(lengthOfObject Damage1 (Meter 22))
(isa Approach1 GeographicalRegion)
(isa Approach2 GeographicalRegion)
(objectTypeFoundInLocation  Rubble Approach1)
(objectTypeFoundInLocation  Rubble Approach2)
```

Figure 2: Some of the axioms "slurped" by Cyc

This "knowledge slurping" approach proved to be an effective way of acquiring knowledge; several thousand axioms were acquired in the course of a few weeks.

A planner was implemented using SubL, a Lisp-like language that underlies Cyc, which made use of these axioms and of other axioms describing key terrain, the location of tanks, trucks and engineering units, and so on. The planner was divided into two modules: a "hypothesize" module and a "test & repair" module. The first module hypothesizes a desired state, looks for the preconditions of that state, then it checks if these preconditions are satisfied; if not, then for each unsatisfied precondition, it recursively hypothesizes actions to fulfil that precondition. This approach relies on some simplifying assumptions about interactions among preconditions. The second module uses the "ask" function in CYC to check whether the preconditions for a desired state are *implied* by the given facts; if not, it explores sets of preconditions to *make* the preconditions true. This exploration covers both rules that can act, and rules that can change the known set of facts. Once all the sub-sub-sub-goals are proved, the workaround plan is considered to be feasible. In short, the planner used backward chaining to "prove" actions, which were then used to build plans, in a similar fashion to AIAI's HTN planner.

As far as ontology is concerned, TFS discovered that every single one of the axioms that were added to Cyc in order to solve the workaround challenge problem inherited – and used – some relevant axioms from Cyc's (pre-existing) upper ontology. It can therefore be claimed that Cyc's existing ontology contributed significantly to the reasoning required for the workarounds challenge problem. However, a fair amount of time was spent defining an intermediate ontology to represent concepts relevant to battle and a "battlespace", such as *spans-Bridgelike*; even more time was spent creating ontological terms that were specific to the challenge problem, such as "the weight of an unladen M88 tracked vehicle". These definitions had to be created before the relevant knowledge could be "slurped". The effort spent on ontology development had an adverse effect on the development of the planner, so the TFS planner that was used to tackle the challenge problems was not able to handle every aspect of the domain.

ISI's approach: EXPECT and knowledge acquisition scripts

ISI, who were part of SAIC's integration team, took a different approach to the workarounds challenge problem. The focus of their work was on using ISI's EXPECT tool as a framework for ontology representation, as a knowledge base for knowledge acquisition support, and as a reasoning tool.

In EXPECT (Gil & Melz 1996; Swartout & Gil, 1996), both factual knowledge and problem-solving knowledge about a task are represented explicitly. This means that the system can access and reason about the representations of factual and problem-solving knowledge and about their interactions. Factual knowledge represents concepts, instances, relations, and the constraints among them. Knowledge is represented in Loom (MacGregor, 1999), a knowledge representation system of the KL-ONE family based on description logic. Every concept or class can have a definition that intensionally describes the set of objects that belong to that class; relations can also have definitions. Loom uses these definitions to produce a subsumption hierarchy that organises all the concepts according to a class/subclass relationship.

Problem solving knowledge is represented in a procedural language that is tightly integrated with the Loom representations. Sub-goals that arise during problem solving are solved by methods. Each method description specifies:

1. The goal that the method can achieve
2. The type of result that the method returns
3. The method body, containing the procedure that must be followed to achieve the method's goal.

Given these capabilities, and the availability of suitable ontologies in Loom, ISI were able to use

EXPECT to perform much of the reasoning needed for workarounds planning; some of the more complex aspects of the problem, such as action selection, required extension of EXPECT's capabilities with a Powerloom-based partial matcher. The ontologies used included the shared HPKB ontology, problem solving method ontologies, domain ontologies and situation information. While EXPECT needed extension to deal with some of the planning tasks, it was able to reason with the large ontologies needed without a significant effect on its performance; further efficiency gains were obtained by compiling the problem solvers.

Perhaps the biggest benefits of using EXPECT for the workarounds challenge problem arose from its ability to critique knowledge, and its use as a knowledge acquisition tool. By analysing the information needed by problem solving methods, EXPECT was able to detect over-generalisations in ontologies (e.g. use of the term "geographical region" where "city" was more appropriate), assumptions in ontologies that were unjustified in this domain (e.g. that objects can only be in one place), and missing information (e.g. unspecified bridge lengths). These capabilities enabled ISI to develop a consistent knowledge base more quickly than would otherwise have been the case.

EXPECT also made use of an approach known as *knowledge acquisition scripts* (Gil & Tallis, 1997) to assist with modification of problem solving knowledge. KA scripts are designed to help users modify the knowledge base in a structured manner; for example, if a problem solving method existed for calculating round trip time for ships, a KA script could assist the knowledge engineer in generalising that procedure to make it applicable for all vehicles. An example of a script (taken from (Gil & Tallis 1997)) can be seen in Figure 3.

```
Applicable when
       (a) A change has caused argument A of a goal G to become more
          general, resulting in goal G-new
       (b) Goal G was achieved by method M before A changed
       (c) G-new can be decomposed into disjunctive subgoals G-1 G-2
       (d) G1 is the same as G

Modification sequence
       CHOICE 1: Create new method M-new based on existing method
              (1) System proposes M as the existing method to be used as a
                  basis. User chooses M or another method.
              (2) System proposes a draft version of M-new that modifies A
                  to match G2. User can make any additional substitution
                  needed in the body of M-new.
              (3) User edits body of M-new if modifications other than
                  substitution are needed
       CHOICE 2: Create new method M-new from scratch

Description of what this KA-Script does:
       Create a method that achieves goal G2 based on method M.

Reasons why it is relevant to the current situation:
       Method M was used before to achieve goal G, which was generalised
       to become the unmatched goal G-new. M may be used to create a new
       method that achieves the other subgoal in this decomposition.
```

Figure 3: A KA script to resolve error type: "Goal G-new cannot be matched"

To summarise, EXPECT appears to be fully capable of reasoning about declarative information and ontologies for planning, as well as performing some knowledge-based planning tasks. The biggest contribution of EXPECT to the workarounds challenge problem was probably its aid for rapid development of knowledge bases, both through critiquing of ontologies and domain knowledge, and through the use of KA scripts to assist modification of problem solving knowledge.

GMU's approach: Collaborative Apprenticeship Multi-strategy Learning

The Learning Agents Lab at George Mason University provided the fourth solution to the workaround planning challenge problem. Their approach is based on the Disciple Toolkit (Tecuci 98; Tecuci et al. 99). The foundation of the Disciple Toolkit is an integration of apprenticeship and multi-strategy learning methods within the Plausible Version Space paradigm. This paradigm allows an expert to teach the agent in much the same way in which the expert would teach a human apprentice - by giving the agent specific examples of tasks and solutions, providing explanations of these solutions, and supervising the agent as it performs new tasks. During such interactions, the expert shares his expertise with the agent, which is continuously extending and improving its knowledge and performance abilities. These kinds of agent capabilities are achieved by a synergistic integration of several learning and knowledge acquisition methods: systematic elicitation of knowledge, empirical inductive learning from examples, learning from explanations, and learning by analogy and experimentation.

The interactions that take place within the Disciple Toolkit are illustrated in Figure 4. This diagram shows that the expert interacts with the system in four ways: eliciting knowledge, helping the system learn rules (from examples), refining and generalising the rules, and handling exceptions. From the examples and the rules, Disciple generates solutions; knowledge base refinement consists of critiquing these solutions, while exception handling deals with the inconsistencies in the knowledge base. The rationale for this approach is that it is easier for experts to update an ontology than to create an ontology; easier to supply examples than to supply rules; easier to understand a sentence in a formal language than to create such a sentence; and easier to give hints than to give explanations. The tight integration of various recognised techniques within Disciple creates a whole system that is arguably more useful and usable than the sum of its parts, and the ease of use of Disciple is expected to lead to rapid acquisition of knowledge from an expert.

For the workaround planning challenge problem, the editing and browsing modules were used to build an ontology describing bridges, river segments, army units and their equipment; the learning module was used to learn rules for destroyed bridges from concrete examples of workarounds and their explanations; and the refinement module was used to generalise or specialise the learned rules, based on the evaluation of workaround scenarios generated by the user. A hierarchical nonlinear planner based on task reduction was also developed and integrated into Disciple to solve workaround problems; an example of a learned task reduction rule can be seen in Figure 5. GMU were fortunate to have an ex-military man on their staff, who acted as the primary user of Disciple (i.e. as a domain expert) during acquisition of workarounds data.

It can be seen that the knowledge acquisition, ontology, and reasoning are much more tightly integrated in Disciple than in the other workaround planners. Disciple's primary aim is to be good at knowledge acquisition, although it is capable of ontology representation and reasoning as well.

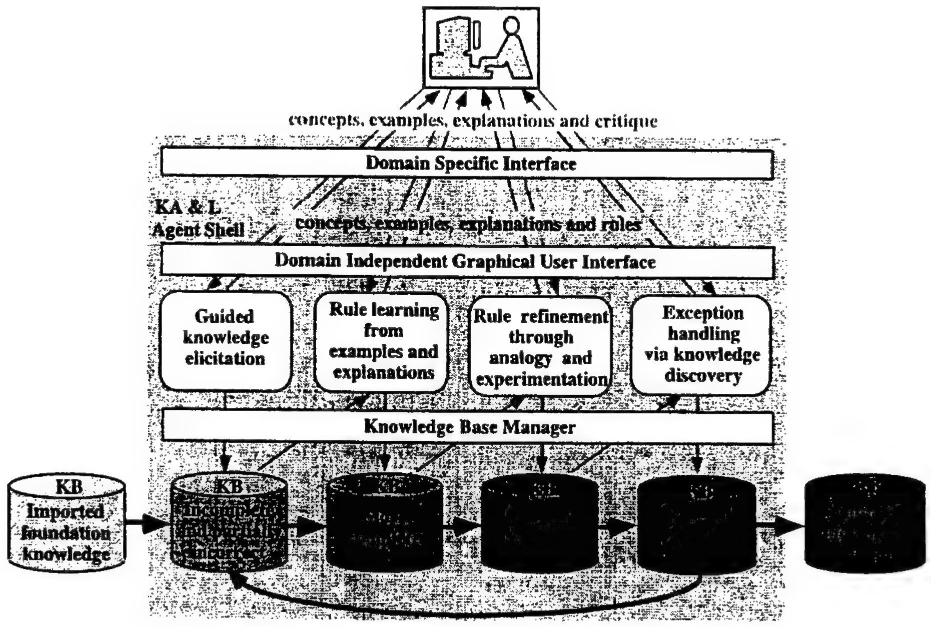

concepts, examples, explanations and critique
Domain Specific Interface
KA & L Agent Shell
concepts, examples, explanations and rules
Domain Independent Graphical User Interface
Guided knowledge elicitation
Rule learning from examples and explanations
Rule refinement through analogy and experimentation
Exception handling via knowledge discovery
Knowledge Base Manager
KB Imported foundation knowledge
KB

Figure 4: Knowledge acquisition and learning processes in Disciple.

Challenge Problem Evaluation

The challenge problem evaluation was carried out June 1998. The format was as follows:

1. A set of questions was issued, that drew on the knowledge made available to everyone in the course of the year. Technology developers were given a short time to generate answers to these questions and mail them to the challenge problem developers (*Test Phase: the test*)
2. Technology developers were given a week or so to improve their systems in the light of their performance on the first set of test questions. They were then allowed to re-submit answers to the first set of test questions. (*Test Phase: the re-test)*
3. Further knowledge was issued that had not previously been available (specifically, knowledge about craters in roads, with definitions of craters and associated attributes). This tested the ability of the systems to enter new knowledge quickly. Five problems that made use of this knowledge were issued simultaneously, and technology developers answered as many as they could *(Modification phase: the test)*
4. After another week, a second set of test questions was issued that concerned working around craters, to test the systems' ability to reason on that new knowledge. Technology developers supplied answers to the five problems already given plus answers to five new problems (*Modification phase: the re- test*).

The answers to the questions were scored for scope (how many solutions, out of all those identified by the senior expert, were found) and score (how accurate the solutions were). Accuracy was scored on five dimensions: correctness of the overall time estimate; viability of the enumerated workaround options; correctness of solution steps provided for each viable option; correctness of temporal constraints among these steps; and appropriateness of engineering resources employed.

```
IF the task to accomplish is
          USE-FIXED-BRIDGE-OVER-BRIDGE-GAP-WITH-MINOR-PREPARATION
                    FOR-BRIDGE          ?O1
                    FOR-GAP-LENGTH  ?N1
                    BY-UNIT                 ?O2
                    WITH-BR-EQ           ?O3
          condition
                    ?O1         IS          BRIDGE
                    ?O2         IS          MILITARY-UNIT
                                HAS-UPPER-ECHELON-EQUIPMENT  ?O4
                                HAS-UPPER-ECHELON-EQUIPMENT  ?O5
                    ?O3         IS          MILITARY-MOBILE-BRIDGE-EQ
                    ?O4         IS          BREACHING-EQ-SET
                                COMPONENT-TYPE  ?O3
                                EQUIPMENT-OF     ?O6
                    ?O5         IS          RUBBLE-CLEARING-EQ-SET
                                EQUIPMENT-OF     ?O6
                    ?O6         IS          MILITARY-UNIT
                                LOCATED-AT         ?O7
                    ?O7         IS          SITE
                    ?N1         IS-IN      (0  1000)
          except when
                    ?O2         HAS-EQUIPMENT  ?O8
                    ?O8         IS          BREACHING-EQ-SET
                                COMPONENT-TYPE  ?O3
          except when
                    ?O2         HAS-EQUIPMENT  ?O9
                    ?O9         IS          RUBBLE-CLEARING-EQ-SET

THEN accomplish the subtasks

?T1       OBTAIN-BRIDGE-AND-PREPARATION-EQUIPMENT-FROM-SAME-UNIT-
          THROUGH-UPPER-ECHELON
                    FOR-BR-EQ-SET       ?O4
                    FOR-PREP-EQ-SET  ?O5
                    FROM-UNIT             ?O6
                    BY-UNIT                 ?O2
                    AT-LOCATION          ?O1

?T2       INSTALL-FIXED-BRIDGE-OVER-BRIDGE-GAP-WITH-MINOR-PREPEPARATION-
          AND-COLOCATED-BRIDGE-AND-PREPARATION-EQUIPMENT
                    FOR-BRIDGE            ?O1
                    FOR-GAP-LENGTH   ?N1
                    WITH-BR-EQ-SET     ?O4
                    WITH-RC-EQ-SET     ?O5
                    AT-LOCATION-EQ   ?O7
                    FOR-UNIT               ?O2
```

Figure 5: A rule learned by the Disciple Toolkit for workaround planning

Results

The results are shown in Figures 6 and 7. To help in understanding the diagrams, AIAI's results will be explained in more detail.

It would be helpful to specify the amount of time available to each technology developer; while precise figures are not available, most of the technology developers were unable to work on the finalised version of the challenge problem until a couple of months before the testing phase (or in AIAI's case, a couple of weeks!) due to various administrative and co-ordination difficulties. While this time period was shorter than expected, it does provide a good estimate of how much knowledge can be captured and reasoned with in a short time period, which was one of the original aims of the HPKB program.

Test Phase: Scope AIAI's system could only answer questions related to bridging of gaps, due to the short development time. The system was only able to provide answers to two of the ten challenge problem questions. Its scope was therefore 20%.

Test Phase: Score AIAI's system initially lost accuracy marks for three reasons: in two cases, a possible solution option had been missed (again, this was a scope problem – the omitted solutions concerned types of bridges that AIAI's system didn't cover), and in one case, a calculation of the total time for a workaround was inaccurate. These problems were fixed, so that by the time of the re-test, the score had reached 100%.

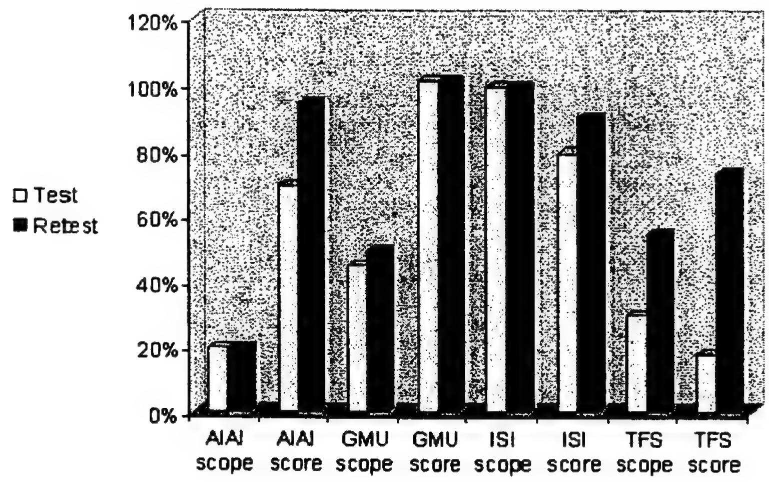


Figure 6: Test phase: test and re-test scope and scores

Overall, it can be seen that ISI and GMU significantly out-performed AIAI and Cycorp/ Teknowledge (TFS), with ISI having a very wide scope and GMU obtaining very high scores. Since an increase in scope is actually likely to lead to a decrease in scores (because the more questions that are answered, the more chance there is to make mistakes), the ISI and GMU systems should be considered "joint winners" of the evaluation. This trade-off is particularly clear in the modification phase, which should provide a truer reflection of the capability of each system, since each technology developer had exactly the same amount of time to make modifications to the system; as Figure 6 shows, as the scope of GMU's system went up, its accuracy slightly decreased.

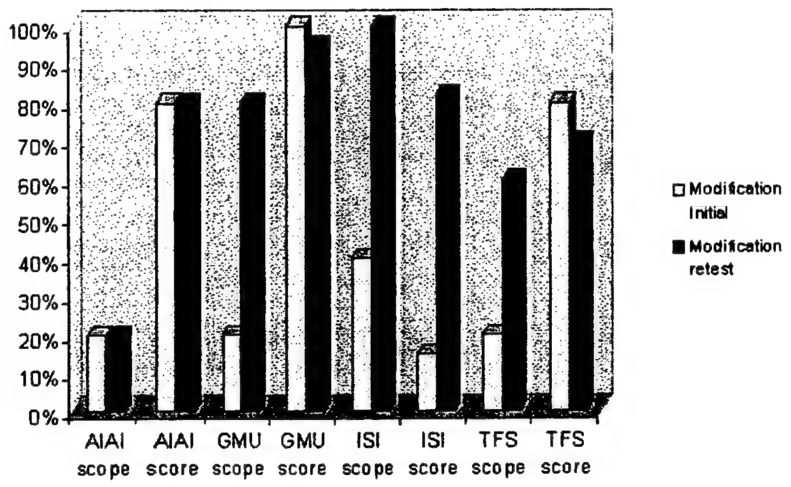


Figure 7: Modification phase: test and re-test scope and scores

The modification phase was also intended to demonstrate the capability of systems for rapid knowledge acquisition. AIAI continued to be handicapped by the narrowness of their previously acquired knowledge, but the other technology developers all achieved significant increases in scope during the modification week. This suggests that the goal of developing systems that can be used to build very large knowledge bases rapidly has been achieved, or (at least) can be achieved in this domain of knowledge.

Strengths of each approach

What were the strengths of each approach?

- **AIAI:** the strength of this system lay in its well-justified ontology of planning, which provided the ability to achieve 100% accurate answers to challenge problem questions – i.e. to generate accurate and fully detailed plans. The fact that such an ontology could not only be built in Cyc, but could also be reasoned about, holds out hope that the reasoning capabilities of Cyc could be significantly extended by further definition of rich ontologies, and of corresponding problem solving methods; see (Sklavakis & Aitken, 1999) for an example of implementation of a problem solving method in Cyc.

- **TFS/Cycorp:** The greatest strength of this system was the ontology framework supplied by Cyc; this provided a noticeable speed-up in acquisition of domain knowledge through knowledge re-use. This effect is shown by the fact that only 35 new concepts had to be developed during the testing fortnight, compared against over 4000 pre-existing assertions that were accessed in one way or another, plus over 1000 assertions regarding vehicles and weapons that were re-used from the ontology developed for the Crisis Management challenge problem. This challenge problem therefore justified Cycorp's claim that a wide-ranging common-sense knowledge base, with a well-determined ontology, supports re-use of knowledge.

- **ISI:** The twin strengths of ISI's system were EXPECT, which can reason about ontologies and their contents as well as performing knowledge based reasoning, and the knowledge acquisition extensions that allowed rapid and accurate modification of the knowledge base (KA-Scripts). This project demonstrated that EXPECT can reason well with a large and rapidly growing ontology. It also demonstrated that EXPECT, with some assistance from Powerloom, was able to reason about the whole domain of knowledge, from the simplest deductions to the most complex calculations. Indeed, some of the calculations regarding Bailey bridges were so complicated that one member of ISI suggested to me (as a representative of the only UK organisation on the HPKB program) that only the British could have invented something like that!

- **GMU:** The rapid development of GMU's system highlighted its integrated knowledge acquisition tool as being its greatest strength. Over the fortnight of testing, GMU added 150 concepts, 100 tasks and 100 problem-solving rules to their knowledge base, representing a 20% increase in concepts, a 100% increase in tasks and a 100% increase in rules. This rate of knowledge acquisition suggests that GMU's system may indeed be able to achieve one of the Holy Grails of knowledge acquisition: rapid, accurate and direct knowledge entry by an expert without intervention from a knowledge engineer. GMU's system was also capable of reasoning about most aspects of the workarounds problem – indeed, it generated a few (correct) solutions that had not been considered by the expert.

Summary

To summarise the lessons to be learned from this paper:

- Rapid development and implementation of very large knowledge bases for planning problems of medium complexity (i.e. plans with about 20 plan steps and 3-4 options for each step) is feasible with current AI technology.
- A knowledge acquisition tool is of great benefit in rapid knowledge base development, whether it is used by the expert to input knowledge or whether it transforms knowledge from other online sources.
- A well-organised and well-justified ontology contributes to both knowledge re-use and a high rate of acquisition of domain knowledge.
- Ontologies of problem solving, and problem solving methods, can provide a big improvement in accuracy of decision making.

References

1. (Aitken & Kingston, 1999) Aitken S. and Kingston J. (1999). *Implementing a Workarounds Planner in Cyc: An HTN Approach*, Report to the DARPA/HPKB project
2. (Cohen et al, 1998) Cohen P., Schrag R., Jones E., Pease A., Lin A., Starr B., Gunning D. and Burke M. (1998). *The DARPA High-Performance Knowledge Bases Project*. AI Magazine, Winter 1998, 25-49.
3. (Cottam et al, 1995) Cottam H., Shadbolt N., Kingston J., Beck H. and Tate A. (1995) *Knowledge Level Planning in the Search and Rescue Domain*, in Research and Development in Expert Systems XII, proceedings of BCS Expert Systems'95, Cambridge, 11-13 December 1995.

4. (Gil & Melz, 1996) Gil Y. and Melz E. (1996) *Explicit Representations of Problem-Solving Strategies to Support Knowledge Acquisition.* In Proceedings of the Thirteen National Conference on Artificial Intelligence (AAAI-96), Portland, Oregon, August 4-8 1996.
5. (Gil & Tallis, 1997) Gil Y. and Tallis M. (1997) *A Script-Based Approach to Modifying Knowledge Bases.* Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97), Providence, RI, July 27-31 1997.
6. (Kingston et al, 1996) Kingston J., Shadbolt N. and Tate A. (1996). *CommonKADS Models for Knowledge Based Planning*, Proceedings of AAAI'96, Portland, Oregon, August 4-8 1996.
7. (Lenat, 1998) Lenat D. (1998). *The Dimensions of Context Space.* Available from http://www.cyc.com/publications.html
8. (Macgregor, 1999) Macgregor R. (1999). *Retrospective on Loom.* http://www.isi.edu/isd/LOOM/papers/ macgregor/Loom_Retrospective.html
9. (Sklavakis & Aitken, 1999) Sklavakis D. and Aitken S. (1999*). Implementing Problem-solving Methods in Cyc.* In Proceedings of IJCAI-99, Stockholm, Sweden, August 1999. AAAI press.
10. (Swartout et al, 1996) Swartout B., Patil R., Knight K. and Russ T. (1996). *Towards Distributed Use of Large-Scale Ontologies.* In Proceedings of KAW '96, Banff, Canada, November 1996. http://ksi.cpsc.ucalgary.ca/KAW/KAW96/swartout/ Banff_96_final_2.html
11. Swartout W.R. and Gil Y.(1996). *EXPECT: A User-Centered Environment for the Development and Adaptation of Knowledge-Based Planning Aids.* In Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative, ed. Austin Tate. Menlo Park, Calif.: AAAI Press, 1996.
12. Tate A, Drabble B. and Dalton J. (1996*). O-Plan: A Knowledge-Based Planner and its Application to Logistics.* In Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative, ed. Austin Tate. Menlo Park, Calif.: AAAI Press, 1996.
13. Tecuci G. (1998). *BUILDING INTELLIGENT AGENTS: An Apprenticeship Multistrategy Learning Theory, Methodology, Tool and Case Studies.* San Diego: Academic Press, 1998.
14. Tecuci G., Boicu M., Wright K., Lee S.W., Marcu D., and Bowman M. (1999). *An Integrated Shell and Methodology for Rapid Development of Knowledge-Based Agents.* In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), July 18-22, Orlando, Florida: AAAI Press, Menlo Park, CA. 1999.

I HTN Planner: CycL Definitions

The CycL definitions for the final (post-repair) version of the HTN planner are listed below. This is deliverable P1.

```
; htn-ketext-challenge-repair
; Stuart Aitken, AIAI, 22 June 98
; Cyclist Jsa

; HTN-style planning rules for workaround
; now with resources

; HTN-style planning, where plan schema are represented as relations:
; The predicate potentialAction holds of an Action, its
; Conditions, the Effects, the Resource requirements,
; and the beginning plan node, and the end plan node.
; Primitive actions have Conditions and Effects and can occur between
; any two plan nodes.
; Composite actions are implemented as rules.
; Composite actions may generalise several primitive actions.
; Composite actions can also have additional Conditions and Effects.
; E.g. (implies (potentialAction prim-action ?C ?E ?R (?I ?J))
;               (potentialAction comp-action (conj ?C C1) ?E ?R (?I ?J)))
; where C1 is an additional condition required by the composite action
; Composite actions may be decomposed into several (composite) sub-actions,
; sub-actions may be supervised (all their preconditions are supplied
; by previous actions) or unsupervised (an unnamed action is added to
; the schema)
; E.g. (implies (and (potentialAction action1 ?C ?X ?R1 (?I ?J))
;                    (potentialAction action2 ?X ?E ?R2 (?J ?K)))
;               (potentialAction comp-action1 ?C ?E * (?I ?K))) *simplified*
; defines that action1 is followed by action2, as ?J is the end/start
; node, and action1 provides the conditions for action2, as ?X is
; shared.
; Unnamed actions can be introduced:
;      (implies (and (isa ?Action PlanAction)
;                    (potentialAction ?Action ?B ?C ?R1 (?H ?I))
;                    (potentialAction action1 ?C ?X ?R2 (?I ?J))
;                    (potentialAction action2 ?X ?E ?R3 (?J ?K)))*simplified*
;               (potentialAction comp-action1 ?B ?E *   (?H ?K))
; Again a simple node sequence is assumed.
; Node networks are actually specified using the (node) constructor to generate
; unique node names. The format is (node <Name> ?BeginNode ?EndNode),
; and this can be used to define several intermediate plan nodes
; between begin and end.

; Conditions and effects are PlanTerms, combined with (conj) as
; opposed to (and), as (and) applies to CycFormula.
; Conditions and effects specified by (CSVCondition), of type:PlanTerm.
; (CSVCondition) applies to a Class, an Instance, a Predicate, and a
; testCSV PlanTerm. testCSV terms consist of a Test-predicate and
; another class instance. The (CSVCondition) is 'true' when the test
; predicate is true of the instances and the Predicate. E.g.
; (CSVCondition ?Class ?X objectFoundInLocation (testCSV equals ?Y))))
; is true when the objectFoundInLocation ?X (of ?Class) is equal to the
; objectFoundInLocation ?Y
; The meaning of each Predicate/Test-predicate combination is defined
; on a case by case basis. A (conjunction of) CSVConditions are true at the
; beginning of planning according to what is true in the BaseKB.
; A CSVCondition replaces a conjuction of KB predicates - it is more
; concise, but (in theory) just definitionally equivalent.

; Resource requirements are statements of the form (allocate <instance>)
; and (consume <resource-type> <amount>)
; plan schema are assumed not to contain allocation or production/consumption
; constraint violations

; Planning is then backward chaining from super to sub-actions, with
; conditions and effects being instantiated or matched as backward
; chaining goes on.

; This approach separates PlanNodes from TemporalThings in general (Cyc should
; not look at all temporal sub-intervals it knows about during a
; constrained planning task). The plan conditions/effects are not
; expressed by holdsIn for similar reasons. In fact, there is no
; explicit construction of what is true in a node.



; ** important note: the potentialAction predicate should not have ***
; ** unbound variables other than in the second argument position ***
; ** else a Cyc crash is likely !! ***
; ** the latest version of Cyc should not have this problem ***


Default Mt: TFPlannerMT.

Constant: TFObject.
F: (isa TFObject Collection).
F: (genls TFObject Thing).

Constant: PlanObjectType.
F: (isa PlanObjectType Collection).
F: (genls PlanObjectType TFObject).

Constant: PlanResourceType.
F: (isa PlanResourceType Collection).
F: (genls PlanResourceType PlanObjectType).

Constant: PlanObject.
F: (isa PlanObject Collection).
F: (genls PlanObject TFObject).

Constant: PlanResource.
F: (isa PlanResource Collection).
F: (genls PlanResource PlanObject).

Constant: PlanResourceConsumable.
F: (isa PlanResourceConsumable Collection).
```

```
F: (genls PlanResourceConsumable PlanResource).

Constant: PlanResourceNonSharable.
F: (isa PlanResourceNonSharable Collection).
F: (genls PlanResourceNonSharable PlanResource).

Constant: MobileEquipmentType.
F: (isa MobileEquipmentType Collection).
F: (genls MobileEquipmentType PlanResourceType).

Constant: TFPredicate.
F: (isa TFPredicate Collection).
F: (genls TFPredicate  TFObject).
F: (genls TFPredicate Predicate).

Constant: WorldConditionPredicate.
F: (isa WorldConditionPredicate  Collection).
F: (genls WorldConditionPredicate TFPredicate).

Constant: TestPredicate.
F: (isa TestPredicate  Collection).
F: (genls TestPredicate TFPredicate).

Constant: PlanAction.
F: (isa PlanAction Collection).
F: (genls PlanAction TFObject).

Constant: PlanAction-Primitive.
F: (isa PlanAction-Primitive Collection).
F: (genls PlanAction-Primitive PlanAction).

Constant: PlanTerm.
F: (isa PlanTerm Collection).
F: (genls PlanTerm TFObject).

Constant: PlanNode.
F: (isa PlanNode Collection).
F: (genls PlanNode  TFObject).
F: (genls PlanNode TemporalThing).

Constant: PlanNodePair.
F: (isa PlanNodePair Collection).
F: (genls PlanNodePair  TFObject).


Constant: BeginNode.
F: (isa BeginNode PlanNode).
Constant: EndNode.
F: (isa EndNode PlanNode).
Constant: Node-1.
F: (isa Node-1 PlanNode).
Constant: Node-2.
F: (isa Node-2 PlanNode).
Constant: Node-3.
F: (isa Node-3 PlanNode).
```

```
Constant: Node-4.
F: (isa Node-4 PlanNode).
Constant: Node-5.
F: (isa Node-5 PlanNode).

Constant: node. ; constructor for node names
F: (isa node NonPredicateFunction).
F: (arity node 3).
F: (arg1Isa node PlanNode).
F: (arg2Isa node PlanNode).
F: (arg3Isa node PlanNode).
F: (resultIsa  node PlanNode).

Constant: pair. ; constructor for pairs of nodes
F: (isa pair NonPredicateFunction).
F: (arity pair 2).
F: (arg1Isa pair PlanNode).
F: (arg2Isa pair PlanNode).
F: (resultIsa pair PlanNodePair).

Constant: holdsInNode. ; a specialisation of holdsIn
F: (isa holdsInNode BinaryPredicate).
F: (arity holdsInNode 2).
F: (arg1Isa holdsInNode PlanNode).
F: (arg2Isa holdsInNode PlanTerm).

; potentialAction is a relation between actions, conditions, effects,
; and plan nodes, this predicate is used to represent tf-style action schemas
Constant: potentialAction.
F: (isa potentialAction QuintaryPredicate).
F: (arity potentialAction 5).
F: (arg1Isa potentialAction PlanAction).  ; action
F: (arg2Isa potentialAction PlanTerm).   ; conditions on applic. of action
F: (arg3Isa potentialAction PlanTerm).   ; effects
F: (arg4Isa potentialAction PlanTerm).   ; resource constraints
F: (arg5Isa potentialAction PlanNodePair).   ; begin-end node pair

Constant: CSVCondition. ; a constructor for PlanTerms
F: (isa CSVCondition NonPredicateFunction).
F: (arity CSVCondition 4).
F: (arg1Isa CSVCondition PlanObjectType).; Class - PlanObjectType
F: (arg2Isa CSVCondition Thing).         ; instance of Class - constrain ?
F: (arg3Isa CSVCondition WorldConditionPredicate). ; WorldConditionPredicate
F: (arg4Isa CSVCondition PlanTerm).      ; testCSV term
F: (resultIsa CSVCondition PlanTerm).


Constant: testCSV. ; a constructor for PlanTerms
F: (isa testCSV NonPredicateFunction).
F: (arity testCSV 2).
F: (arg1Isa testCSV TestPredicate).      ; TestPredicate
F: (arg2Isa testCSV Thing).              ; instance- can't constrain
F: (resultIsa testCSV PlanTerm).

F: (isa True PlanTerm).
```

```
Constant: consume.  ; a constructor for PlanTerms
F: (isa consume NonPredicateFunction).
F: (arity consume 2).
F: (arg1Isa consume PlanResourceType). ; TYPE of resource consumed
F: (arg1Genl consume PlanResourceConsumable). ; must be consumable
F: (arg2Isa consume Thing).            ; amount consumed or instance of type
F: (resultIsa consume PlanTerm).

Constant: allocate.  ; a constructor for PlanTerms
F: (isa allocate NonPredicateFunction).
F: (arity allocate 1).
F: (arg1Isa allocate PlanResource).    ; a specific resource is allocated
F: (resultIsa allocate PlanTerm).

Constant: conj. ; a constructor for PlanTerms = logical 'and'
F: (isa conj NonPredicateFunction).
F: (arity conj 2).
F: (arg1Isa conj PlanTerm).
F: (arg2Isa conj PlanTerm).
F: (resultIsa conj PlanTerm).

Constant: achievableAction.
F: (isa achievableAction BinaryPredicate).
F: (arity achievableAction 2).
F: (arg1Isa achievableAction PlanAction).
F: (arg2Isa achievableAction PlanTerm).

; define some specific predicates
; (objectFoundInLocation Site Place) but don't make assumptions about
; the types of Site and Place

Constant: lessThan.
F: (isa lessThan BinaryPredicate).
F: (arity lessThan 2).
F: (arg1Isa lessThan ScalarInterval).
F: (arg2Isa lessThan ScalarInterval).

Constant: bridgedBy.
F: (isa bridgedBy BinaryPredicate).
F: (arity bridgedBy 2).
F: (arg1Isa bridgedBy Thing).
F: (arg2Isa bridgedBy Thing).

Constant: Some.
F: (isa Some Thing).

;-------------------------------------------------------------

; Define the CSV Conditions. In general, this cannot be done once for
; all Predicates and Classes, but is defined specifically for each
; combination
; all rules are really equivalences, but only define one direction for now,
; and these rules are true at BeginNode, i.e.the plan node that
; represents the current state of the world
; true in BaseKB now ==> true in BeginNode

;;; define the predicates used in planning to be of the appropriate type
F: (isa lessThan TestPredicate).
F: (isa equals TestPredicate).

F: (isa bridgedBy WorldConditionPredicate).
F: (isa objectFoundInLocation WorldConditionPredicate).
F: (isa gapLength WorldConditionPredicate).
F: (isa leftBankSlope WorldConditionPredicate).
F: (isa rightBankSlope WorldConditionPredicate).
F: (isa leftBankSurfaceAttr WorldConditionPredicate).
F: (isa rightBankSurfaceAttr WorldConditionPredicate).


; if a Class has an instance at Loc:
F: (implies (and (classHasInstanceAt ?Class ?Loc)
 (isa ?Class PlanResourceType))
           (holdsInNode BeginNode
           (CSVCondition ?Class Some objectFoundInLocation
      (testCSV equals ?Loc)))).

F: (implies (and (classHasInstanceAt ?Class ?Loc)
 (isa ?Class PlanResourceType)
 (leftRegion ?CS ?Loc))
           (holdsInNode BeginNode
           (CSVCondition ?Class Some objectFoundInLocation
      (testCSV equals (leftRegion-Fn ?CS))))).

F: (implies (and (classHasInstanceAt ?Class ?Loc)
 (isa ?Class PlanResourceType)
 (rightRegion ?CS ?Loc))
           (holdsInNode BeginNode
           (CSVCondition ?Class Some objectFoundInLocation
      (testCSV equals (rightRegion-Fn ?CS))))).



; gapLength ?LSite lessThan ?X if ?X >= ?LSite
F: (implies (and (gapLength ?Site (Meter ?LSite))
                 (regionIsa ?Site SpanSite)
         (greaterThanOrEqualTo ?X ?LSite))
           (holdsInNode BeginNode
           (CSVCondition Site ?Site gapLength (testCSV lessThan (Meter ?X))))).

; a site is bridged if:
F: (implies (and (isa ?Site SpanSite)
                 (bridgedBy ?Site ?X))
           (holdsInNode BeginNode
    (CSVCondition Site ?Site bridgedBy (testCSV equals ?X)))).

; leftBankSlope ?LSite lessThan ?X if ?X >= ?LSite
F: (implies (and (leftBankSlope ?Site (Percent ?LSite))
         (greaterThanOrEqualTo ?X ?LSite))
           (holdsInNode BeginNode
    (CSVCondition Site ?Site leftBankSlope
                     (testCSV lessThan (Percent ?X))))).

;F: (CSVCondition Site ?Site leftBankSlope  ;the don't care value
;                     (testCSV lessThan (Percent 500))).

F: (implies (and (rightBankSlope ?Site (Percent ?LSite))
         (greaterThanOrEqualTo ?X ?LSite))
            (holdsInNode BeginNode
    (CSVCondition Site ?Site rightBankSlope
                     (testCSV lessThan (Percent ?X))))).

;F: (CSVCondition Site ?Site rightBankSlope  ;the don't care value
;                     (testCSV lessThan (Percent 500))).


; locations have a surfaceAttr
F: (implies (leftBankSurfaceAttr ?CS ?ST)
            (holdsInNode BeginNode
    (CSVCondition Site ?CS leftBankSurfaceAttr
   (testCSV equals ?ST)))).

F: (implies (rightBankSurfaceAttr ?CS ?ST)
            (holdsInNode BeginNode
    (CSVCondition Site ?CS rightBankSurfaceAttr
   (testCSV equals ?ST)))).


; conjunctions of CSVConditions true in the BaseKB can be built up
; assume there will be at most 5 CSVConditions in the conditions
; of any action, and define these rules non-recursively

F: (implies (and (holdsInNode BeginNode
                     (CSVCondition ?Class ?X ?P (testCSV ?PT ?Y)))
 (holdsInNode BeginNode
                     (CSVCondition ?Class1 ?X1 ?P1 (testCSV ?PT1 ?Y1)))
 (holdsInNode BeginNode
                     (CSVCondition ?Class2 ?X2 ?P2 (testCSV ?PT2 ?Y2)))
 (holdsInNode BeginNode
                     (CSVCondition ?Class3 ?X3 ?P3 (testCSV ?PT3 ?Y3)))
 (holdsInNode BeginNode
                     (CSVCondition ?Class4 ?X4 ?P4 (testCSV ?PT4 ?Y4))))
(holdsInNode BeginNode
                     (conj (CSVCondition ?Class ?X ?P (testCSV ?PT ?Y))
 (conj (CSVCondition ?Class1 ?X1 ?P1 (testCSV ?PT1 ?Y1))
 (conj (CSVCondition ?Class2 ?X2 ?P2 (testCSV ?PT2 ?Y2))
                     (conj (CSVCondition ?Class3 ?X3 ?P3 (testCSV ?PT3 ?Y3))
       (CSVCondition ?Class4 ?X4 ?P4 (testCSV ?PT4 ?Y4)))))))).
F: (implies (and (holdsInNode BeginNode
                     (CSVCondition ?Class ?X ?P (testCSV ?PT ?Y)))
 (holdsInNode BeginNode
                     (CSVCondition ?Class1 ?X1 ?P1 (testCSV ?PT1 ?Y1)))
 (holdsInNode BeginNode
                     (CSVCondition ?Class2 ?X2 ?P2 (testCSV ?PT2 ?Y2)))
 (holdsInNode BeginNode
                (CSVCondition ?Class3 ?X3 ?P3 (testCSV ?PT3 ?Y3))))
(holdsInNode BeginNode
                (conj (CSVCondition ?Class ?X ?P (testCSV ?PT ?Y))
 (conj (CSVCondition ?Class1 ?X1 ?P1 (testCSV ?PT1 ?Y1))
 (conj (CSVCondition ?Class2 ?X2 ?P2 (testCSV ?PT2 ?Y2))
                   (CSVCondition ?Class3 ?X3 ?P3 (testCSV ?PT3 ?Y3))))))).

F: (implies (and (holdsInNode BeginNode
                (CSVCondition ?Class ?X ?P (testCSV ?PT ?Y)))
 (holdsInNode BeginNode
                (CSVCondition ?Class1 ?X1 ?P1 (testCSV ?PT1 ?Y1)))
 (holdsInNode BeginNode
                (CSVCondition ?Class2 ?X2 ?P2 (testCSV ?PT2 ?Y2))))
(holdsInNode BeginNode (conj
                (CSVCondition ?Class ?X ?P (testCSV ?PT ?Y))
 (conj (CSVCondition ?Class1 ?X1 ?P1 (testCSV ?PT1 ?Y1))
 (CSVCondition ?Class2 ?X2 ?P2 (testCSV ?PT2 ?Y2)))))).

F: (implies (and (holdsInNode BeginNode
                (CSVCondition ?Class ?X ?P (testCSV ?PT ?Y)))
 (holdsInNode BeginNode
                (CSVCondition ?Class1 ?X1 ?P1 (testCSV ?PT1 ?Y1))))
(holdsInNode BeginNode  (conj
                (CSVCondition ?Class ?X ?P (testCSV ?PT ?Y))
 (CSVCondition ?Class1 ?X1 ?P1 (testCSV ?PT1 ?Y1))))).

; the conditions and effects of a primitive plan actions can be combined:
; hope that this rule is sufficient - given the aim of
; preserving a (conj P (conj Q R))) structure for conditions
; and effects
; -- will work as long as only one potentialAction relation for a
; particular action has a (conj) term as condition and/or effect
; the others can only have a single (CSVCondition) as condition and as
; effect
; this should be OK as 1 relation defines what the action does, and
; the others define which CSVConditions don't change, these should
; only have 1 (CSVCondition) as condition and 1 effect
; unfortunately, a recursive spec. seems unavoidable

F: (implies (and
 (isa ?Action PlanAction-Primitive)
 (potentialAction ?Action ?C1
(CSVCondition ?Class1 ?I1 ?P1 (testCSV ?PT1 ?Y1)) ?R1
(pair ?I ?J))
 (potentialAction ?Action ?C2 ?E2 ?R2 (pair ?I ?J)))
 (potentialAction ?Action (conj ?C1 ?C2)
(conj (CSVCondition ?Class1 ?I1 ?P1 (testCSV ?PT1 ?Y1))
      ?E2) (conj ?R1 ?R2) (pair ?I ?J))).


F: (implies (potentialAction ?Action ?C (conj ?E1 (conj ?E2 ?E3))
 ?R (pair ?I ?J))
    (potentialAction ?Action ?C (conj ?E2 (conj ?E1 ?E3))
 ?R (pair ?I ?J))).
```

```
F: (implies (potentialAction ?Action (conj ?E1 (conj ?E2 ?E3)) ?E4
 ?R (pair ?I ?J))
    (potentialAction ?Action (conj ?E2 (conj ?E1 ?E3)) ?E4
 ?R (pair ?I ?J))).

F: (implies (potentialAction ?Action (conj ?E1 (conj ?E2 (conj ?E3 ?E4))) ?E5
 ?R (pair ?I ?J))
    (potentialAction ?Action (conj ?E2 (conj ?E3 (conj ?E4 ?E1))) ?E5
 ?R (pair ?I ?J))).

F: (implies (potentialAction ?Action ?C (conj ?E1 (conj ?E2 (conj ?E3 ?E4)))
 ?R (pair ?I ?J))
    (potentialAction ?Action ?C (conj ?E4 (conj ?E1 (conj ?E2 ?E3)))
 ?R (pair ?I ?J))).
F: (implies (potentialAction ?Action ?C (conj ?E1 (conj ?E2 (conj ?E3 ?E4)))
 ?R (pair ?I ?J))
    (potentialAction ?Action ?C (conj ?E2 (conj ?E1 (conj ?E3 ?E4)))
 ?R (pair ?I ?J))).

;------------------------------------------------------------

; now some action relations and rules - action conditions are
; expressed as PlanTerms, as introduced above

; generalising....
; the gap-to-be-spanned < capability-of-bridge is always a condition
; the bank-slope < allowable-bank-slope is a condition for AVLB
; the wetness-of-the-river = Wet is a condition for RibbonBridges
; time is modelled as a resource


; if there are 3 conditions, the required syntax is:
;  (conj (CSVCondition ..) (conj (CSVCondition ..) (CSVCondition ..)))
; and so on.

;------------------------------------------------------------
; begin with the rules that Cyc should try LAST: composite actions
; but first declare the action names

Default Mt: TFPlannerMT.

Constant: spanGap.
F: (isa  spanGap PlanAction).

Constant: useAVLB.
F: (isa  useAVLB PlanAction).

Constant: useMGB.
F: (isa  useMGB PlanAction).

Constant: mobiliseMGB.
F: (isa  mobiliseMGB PlanAction).

Constant: mobiliseAVLB.
F: (isa  mobiliseAVLB PlanAction).
```

```
Constant: narrowGap.
F: (isa narrowGap PlanAction).

Constant: emplaceAVLB.
F: (isa emplaceAVLB PlanAction-Primitive).

Constant: emplaceMGB.
F: (isa emplaceMGB PlanAction-Primitive).

Constant: prepareNearBank.
F: (isa prepareNearBank PlanAction-Primitive).

Constant: prepareFarBank.
F: (isa prepareFarBank PlanAction-Primitive).

Constant: flattenBank-ForAVLB.
F: (isa flattenBank-ForAVLB PlanAction).

Constant: flattenBank-Left.
F: (isa flattenBank-Left PlanAction-Primitive).

Constant: flattenBank-Right.
F: (isa flattenBank-Right  PlanAction-Primitive).

Constant: obtainOpControlDivision.
F: (isa obtainOpControlDivision PlanAction-Primitive).

Constant: obtainOpControlCorps.
F: (isa obtainOpControlCorps PlanAction-Primitive).

Constant: transportEquipment.
F: (isa transportEquipment NonPredicateFunction).
F: (arity transportEquipment 1).
F: (arg1Isa transportEquipment MobileEquipmentType).
F: (resultIsa transportEquipment PlanAction-Primitive).

Constant: bulldozeSoil-ForAVLB.
F: (isa bulldozeSoil-ForAVLB PlanAction-Primitive).

;-------------------------------------------------------------


; composite actions

; spanGap.

; if an AVLB is at the gap site use that:
F: (implies (potentialAction useAVLB ?A ?E ?R (pair ?I ?J))
    (potentialAction spanGap ?A ?E ?R (pair ?I ?J))).

; get an MGB Co
F: (implies (potentialAction useMGB ?A ?E ?R (pair ?I ?J))
    (potentialAction spanGap ?A ?E ?R (pair ?I ?J))).
```

```
;----------------------------------------------------------

; useAVLB

; one decomposition is:
; [narrowGap; emplaceAVLB; prepareFarBank]
F: (implies (and
         (potentialAction prepareFarBank ?E2 ?E ?R3    ;3rd action
(pair (node Node-2 ?I ?J) ?J))
         (potentialAction emplaceAVLB ?E1 ?E2 ?R2      ;2nd action
(pair (node Node-1 ?I ?J) (node Node-2 ?I ?J)))
 (potentialAction narrowGap ?C ?E1 ?R1          ;1st action
(pair ?I (node Node-1 ?I ?J))))
      (potentialAction useAVLB ?C ?E (conj ?R1 (conj ?R2 ?R3))
(pair ?I ?J))).

; another is:
; [prepareNearBank; emplaceAVLB; prepareFarBank]

F: (implies (and
         (potentialAction prepareFarBank ?E2 ?E ?R3    ;3rd action
(pair (node Node-2 ?I ?J) ?J))
         (potentialAction emplaceAVLB ?E1 ?E2 ?R2      ;2nd action
(pair (node Node-1 ?I ?J) (node Node-2 ?I ?J)))
 (potentialAction prepareNearBank ?C ?E1 ?R1    ;1st action
(pair ?I (node Node-1 ?I ?J))))
      (potentialAction useAVLB ?C ?E (conj ?R1 (conj ?R2 ?R3))
(pair ?I ?J))).

; another is:
; [mobiliseAVLB; flattenBank-ForAVLB; emplaceAVLB; prepareFarBank]
F: (implies (and
         (potentialAction prepareFarBank ?E3 ?E ?R4    ;4th action
(pair (node Node-3 ?I ?J) ?J))
         (potentialAction emplaceAVLB ?E2 ?E3 ?R3      ;3rd action
(pair (node Node-2 ?I ?J) (node Node-3 ?I ?J)))
 (potentialAction flattenBank-ForAVLB ?E1 ?E2 ?R2        ;2nd action
(pair (node Node-1 ?I ?J) (node Node-2 ?I ?J)))'
 (potentialAction mobiliseAVLB ?C ?E1 ?R1          ;1st action
(pair ?I (node Node-1 ?I ?J))))
      (potentialAction useAVLB ?C ?E (conj ?R1 (conj ?R2 (conj ?R3 ?R4)))
(pair ?I ?J))).
;----------------------------------------------------------

; useMGB

; one decomposition is:
; [mobiliseMGB; prepareNearBank; emplaceMGB; prepareFarBank]
F: (implies (and
         (potentialAction prepareFarBank ?E3 ?E ?R4    ;4th action
(pair (node Node-3 ?I ?J) ?J))
         (potentialAction emplaceMGB ?E2 ?E3 ?R3      ;3rd action
(pair (node Node-2 ?I ?J) (node Node-3 ?I ?J)))
```

```
 (potentialAction prepareNearBank ?E1 ?E2 ?R2        ;2nd action
(pair (node Node-1 ?I ?J) (node Node-2 ?I ?J)))
 (potentialAction mobiliseMGB ?C ?E1 ?R1          ;1st action
(pair ?I (node Node-1 ?I ?J))))
      (potentialAction useMGB ?C ?E (conj ?R1 (conj ?R2 (conj ?R3 ?R4)))
(pair ?I ?J))).

;-----------------------------------------------------------

; mobiliseAVLB from remote site
; assume a Bulldozer will always be required too

; one decomposition is:
; [obtainOpcontrolDivision; ;;; of the AVLB at Division level
;    (transportEquipment ArmoredVehicleLaunchedBridge) in parallel with
;    (transportEquipment Bulldozer)]
; assume control of Bulldozer is incorporated with Corps level control action

F: (implies (and
         (potentialAction (transportEquipment ArmoredVehicleLaunchedBridge)
?E1 ?E2 ?R2 (pair (node Node-1 ?I ?J) ?J)) ;3rd action
 (potentialAction (transportEquipment Bulldozer) ?E1 ?E3 ?R3 ;2nd action
(pair (node Node-1 ?I ?J) ?J))
 (potentialAction obtainOpControlDivision ?C ?E1 ?R1   ;1st action
(pair ?I (node Node-1 ?I ?J))))
      (potentialAction mobiliseAVLB ?C (conj ?E2 ?E3) (conj ?R1 ?R2)   ;ignore R3
(pair ?I ?J))).

;-----------------------------------------------------------

; mobiliseMGB
; assume a Bulldozer will always be required too

Default Mt: TFPlannerMT.
; one decomposition is:
; [obtainOpcontrolCorps; ;;; of the MGBSet at Corps level
;    (transportEquipment MGBSet) in parallel with (transportEquipment Bulldozer)]
; assume control of Bulldozer is subsumed by Corps level control action

F: (implies (and
         (potentialAction (transportEquipment MGBSet) ?E1A ?E2 ?R2 ;3rd action
(pair (node Node-1 ?I ?J) ?J))
 (potentialAction (transportEquipment Bulldozer) ?E1B ?E3 ?R3 ;2nd action
(pair (node Node-1 ?I ?J) ?J))
 (potentialAction obtainOpControlCorps ?C (conj ?E1A ?E1B) ?R1 ;1st action
(pair ?I (node Node-1 ?I ?J))))
      (potentialAction mobiliseMGB ?C (conj ?E2 ?E3) (conj ?R1 ?R2)   ;ignore R3
(pair ?I ?J))).

;-----------------------------------------------------------

; narrowGap

; one decomposition is:
; [obtainOpcontrolDivision; (transportEquipment Bulldozer); bulldozeSoil-ForAVLB]
; assuming Bulldozers are division level resources

F: (implies (and
        (potentialAction bulldozeSoil-ForAVLB ?E2 ?E ?R3        ;3rd action
(pair (node Node-2 ?I ?J) ?J))
        (potentialAction (transportEquipment Bulldozer) ?E1 ?E2 ?R2  ;2nd action
(pair (node Node-1 ?I ?J) (node Node-2 ?I ?J)))
 (potentialAction obtainOpControlDivision ?C ?E1 ?R1     ;1st action
(pair ?I (node Node-1 ?I ?J))))
      (potentialAction narrowGap ?C ?E (conj ?R1 (conj ?R2 ?R3))
(pair ?I ?J))).

; another:
; [bulldozeSoil-ForAVLB]

;-----------------------------------------------------------

; composite actions which generalise single primitive actions

; flattenBank-ForAVLB

F: (implies (potentialAction flattenBank-Left ?A ?E ?R (pair ?I ?J))
    (potentialAction flattenBank-ForAVLB ?A ?E ?R (pair ?I ?J))).

F: (implies (potentialAction flattenBank-Right ?A ?E ?R (pair ?I ?J))
    (potentialAction flattenBank-ForAVLB ?A ?E ?R (pair ?I ?J))).

;-----------------------------------------------------------


; emplaceAVLB  (previously: spanWithAVLB)

; if an AVLB is at a site with a gap of < 17.37m, and
; bank slope < 13.5 (30%) then the site can be bridged

; case 1 AVLB on left bank
F: (potentialAction emplaceAVLB
(conj  (CSVCondition ArmoredVehicleLaunchedBridge Some
objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site)))
     (conj  (CSVCondition Site ?Site gapLength
  (testCSV lessThan (Meter 17.37)))
        (CSVCondition Site ?Site leftBankSlope
                 (testCSV lessThan (Percent 30)))))
         (CSVCondition Site ?Site bridgedBy (testCSV equals True))
 (consume MilitaryOpTime (MinutesDuration 5 10))
         (pair ?I ?J)).

 ; case 2 AVLB on right bank
 F: (potentialAction emplaceAVLB
 (conj  (CSVCondition ArmoredVehicleLaunchedBridge
  Some objectFoundInLocation
 (testCSV equals (rightRegion-Fn ?Site)))
      (conj  (CSVCondition Site ?Site gapLength
  (testCSV lessThan (Meter 17.37)))
```

```
       (CSVCondition Site ?Site rightBankSlope
                (testCSV lessThan (Percent 30)))))
        (CSVCondition Site ?Site bridgedBy (testCSV equals True))
(consume MilitaryOpTime (MinutesDuration 5 10))
        (pair ?I ?J)).



F: (potentialAction emplaceAVLB
        (CSVCondition ?Class ?B objectFoundInLocation (testCSV equals ?Site))
        (CSVCondition ?Class ?B objectFoundInLocation (testCSV equals ?Site))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

F: (potentialAction emplaceAVLB
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

F: (potentialAction emplaceAVLB
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).


;------------------------------------------------------------


; emplaceMGB

; if an MGB is at a site with a gap of < 31.09m, and

F: (potentialAction emplaceMGB
(conj  (CSVCondition MGBSet Some objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site)))
            (CSVCondition Site ?Site gapLength
 (testCSV lessThan (Meter 31.09))))
        (CSVCondition Site ?Site bridgedBy (testCSV equals True))
(consume MilitaryOpTime (TimesFn (MGBConstructionFn ?Site) 1.15))
        (pair ?I ?J)).

F: (potentialAction emplaceMGB
(conj  (CSVCondition MGBSet Some objectFoundInLocation
(testCSV equals (rightRegion-Fn ?Site)))
            (CSVCondition Site ?Site gapLength
 (testCSV lessThan (Meter 31.09))))
        (CSVCondition Site ?Site bridgedBy (testCSV equals True))
(consume MilitaryOpTime (TimesFn (MGBConstructionFn ?Site) 1.15))
        (pair ?I ?J)).

F: (potentialAction emplaceMGB
        (CSVCondition ?Class ?B objectFoundInLocation (testCSV equals ?Site))
        (CSVCondition ?Class ?B objectFoundInLocation (testCSV equals ?Site))
```

```
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

F: (potentialAction emplaceMGB
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).
F: (potentialAction emplaceMGB
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).
;--------------

Default Mt: TFPlannerMT.

; (transportEquipment ?Class)

F: (potentialAction (transportEquipment ?Class)
(CSVCondition Site ?Site1 gapLength (testCSV lessThan ?X))
(CSVCondition Site ?Site1 gapLength (testCSV lessThan ?X))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

F: (potentialAction (transportEquipment ?Class)
(CSVCondition Site ?Site1 leftBankSlope (testCSV lessThan ?X))
(CSVCondition Site ?Site1 leftBankSlope (testCSV lessThan ?X))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

F: (potentialAction (transportEquipment ?Class)
(CSVCondition Site ?Site1 rightBankSlope (testCSV lessThan ?X))
(CSVCondition Site ?Site1 rightBankSlope (testCSV lessThan ?X))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

F: (potentialAction (transportEquipment ?Class)
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).
F: (potentialAction (transportEquipment ?Class)
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

; this says that all things that are not of ?Class1 are not moved
; actually, only one thing of ?Class1 gets moved so the rule is too general
F: (implies (not (equals ?Class2 ?Class1))
      (potentialAction (transportEquipment ?Class1)
(CSVCondition ?Class2 ?B objectFoundInLocation (testCSV equals ?Site1))
        (CSVCondition ?Class2 ?B objectFoundInLocation (testCSV equals ?Site1))
(consume MilitaryOpTime (MinutesDuration 0))
```

```
      (pair ?I ?J))).


; some instance of ?Class moves from ?Site1 to ?Site2
; whether it goes to the left or right region is undetermined
F: (implies (and (regionIsa ?Site1 FarInterdictionSite)
 (regionIsa ?Site2 NearInterdictionSite))
    (potentialAction (transportEquipment ?Class)
(CSVCondition ?Class ?B objectFoundInLocation
(testCSV equals ?Site1))
        (CSVCondition ?Class ?B objectFoundInLocation
(testCSV equals ?Site2))
(consume MilitaryOpTime
         (TimesFn (DividesFn (DistanceFn ?Site1 ?Site2) 60) 60))
        (pair ?I ?J))).

;;; something can be moved to the left region if this region
;;; is between the far site and the river i.e. it does not need
;;; to cross the river !
F: (implies (and (regionIsa ?Site1 FarInterdictionSite)
 (regionIsa ?Site2 NearInterdictionSite)
 (leftRegion ?Site2 ?LR)
 (isa ?R River)
 (between ?Site1 ?R ?LR))
    (potentialAction (transportEquipment ?Class)
(CSVCondition ?Class ?B objectFoundInLocation
(testCSV equals ?Site1))
        (CSVCondition ?Class ?B objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site2)))
(consume MilitaryOpTime
         (TimesFn (DividesFn (DistanceFn ?Site1 ?Site2) 60) 60))
        (pair ?I ?J))).

F: (implies (and (regionIsa ?Site1 FarInterdictionSite)
 (regionIsa ?Site2 NearInterdictionSite)
 (rightRegion ?Site2 ?RR)
 (isa ?R River)
 (between ?R ?Site1 ?RR))
    (potentialAction (transportEquipment ?Class)
(CSVCondition ?Class ?B objectFoundInLocation
(testCSV equals ?Site1))
        (CSVCondition ?Class ?B objectFoundInLocation
(testCSV equals (rightRegion-Fn ?Site2)))
(consume MilitaryOpTime
         (TimesFn (DividesFn (DistanceFn ?Site1 ?Site2) 60) 60))
        (pair ?I ?J))).


;---------------

; obtainOpControl

F: (potentialAction obtainOpControlDivision ?C ?C
(consume MilitaryOpTime (MinutesDuration 120 180))
(pair ?I ?J)).
```

```
F: (potentialAction obtainOpControlCorps ?C ?C
(consume MilitaryOpTime (MinutesDuration 240 360))
(pair ?I ?J)).

;--------------

Default Mt: TFPlannerMT.

; bulldozeSoil-ForAVLB

; the gapLength can be reduced by up to 10m
;; from the left
F: (potentialAction bulldozeSoil-ForAVLB
  (conj (CSVCondition Bulldozer Some objectFoundInLocation
 (testCSV equals (leftRegion-Fn ?Site1)))
  (conj (CSVCondition Site ?Site1  leftBankSurfaceAttr
   (testCSV equals SoftSurface))
        (CSVCondition Site ?Site1 gapLength
   (testCSV lessThan (PlusFn ?X (Meter 10))))))
  (conj (CSVCondition Bulldozer Some objectFoundInLocation
 (testCSV equals (leftRegion-Fn ?Site1)))
  (conj (CSVCondition Site ?Site1 gapLength
  (testCSV lessThan ?X))
        (CSVCondition Site ?Site1 leftBankSlope
 (testCSV lessThan (Percent 30)))))
 (consume MilitaryOpTime
 (TimesFn (DividesFn  ;AVLBwidth+2 * section of remaining gap
   (TimesFn 10 (CrossSectionFn ?Site1 15)) ; 17 - 2
                             (TimesFn 250 0.75 0.8)) ;rate * factors
                            1.15 60))   ; expected time factor * 60 mins
        (pair ?I ?J)).

;; from the right
F: (potentialAction bulldozeSoil-ForAVLB
  (conj (CSVCondition Bulldozer Some objectFoundInLocation
 (testCSV equals (rightRegion-Fn ?Site1)))
  (conj (CSVCondition Site ?Site1  rightBankSurfaceAttr
   (testCSV equals SoftSurface))
        (CSVCondition Site ?Site1 gapLength
   (testCSV lessThan (PlusFn ?X (Meter 10))))))
     (conj (CSVCondition Bulldozer Some objectFoundInLocation
 (testCSV equals (rightRegion-Fn ?Site1)))
  (conj (CSVCondition Site ?Site1 gapLength
  (testCSV lessThan ?X))
        (CSVCondition Site ?Site1 rightBankSlope
 (testCSV lessThan (Percent 30)))))
 (consume MilitaryOpTime
 (TimesFn (DividesFn  ;AVLBwidth+2 * section of remaining gap
   (TimesFn 10 (CrossSectionFn ?Site1 15)) ; 17 - 2
                             (TimesFn 250 0.75 0.8)) ;rate * factors
                            1.15 60))   ; expected time factor * 60 mins
        (pair ?I ?J)).
```

```
F: (potentialAction bulldozeSoil-ForAVLB
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).
F: (potentialAction bulldozeSoil-ForAVLB
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

F: (potentialAction bulldozeSoil-ForAVLB
(CSVCondition ?Class ?B objectFoundInLocation (testCSV equals ?Site1))
        (CSVCondition ?Class ?B objectFoundInLocation (testCSV equals ?Site1))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

;--------------

; flattenBank-Left

; the leftBankSlope can be reduced to less than 13.5 Degrees
F: (potentialAction flattenBank-Left
  (conj (CSVCondition Bulldozer Some objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site1)))
        (CSVCondition Site ?Site1  leftBankSurfaceAttr
   (testCSV equals SoftSurface)))
  (CSVCondition Site ?Site1 leftBankSlope
(testCSV lessThan (Percent 30)))
(consume MilitaryOpTime
(TimesFn (DividesFn  ;AVLBwidth+2 * section of remaining bank
  (TimesFn 10 (TriSectionFn (leftBankSite-Fn ?Site1))) ;
                            (TimesFn 250 0.75 0.8)) ;rate * factors
                           1.15 60))   ; expected time factor * 60 mins
        (pair ?I ?J)).

F: (potentialAction flattenBank-Left
(CSVCondition Site ?Site1 gapLength (testCSV lessThan ?X))
(CSVCondition Site ?Site1 gapLength (testCSV lessThan ?X))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

; rightBankSlope is not changed
F: (potentialAction flattenBank-Left
(CSVCondition Site ?Site1 rightBankSlope (testCSV lessThan ?X))
(CSVCondition Site ?Site1 rightBankSlope (testCSV lessThan ?X))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

F: (potentialAction flattenBank-Left
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).
```

```
F: (potentialAction flattenBank-Left
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

F: (potentialAction flattenBank-Left
(CSVCondition ?Class ?B objectFoundInLocation (testCSV equals ?Site1))
        (CSVCondition ?Class ?B objectFoundInLocation (testCSV equals ?Site1))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).
;--------------

; flattenBank-Right

; the leftBankSlope can be reduced to less than 13.5 Degrees = 30 % ??
F: (potentialAction flattenBank-Right
  (conj (CSVCondition Bulldozer Some objectFoundInLocation
(testCSV equals (rightRegion-Fn ?Site1)))
        (CSVCondition Site ?Site1  rightBankSurfaceAttr
   (testCSV equals SoftSurface)))
  (CSVCondition Site ?Site1 rightBankSlope
(testCSV lessThan (Percent 30)))
(consume MilitaryOpTime
(TimesFn (DividesFn  ;AVLBwidth+2 * section of remaining bank
  (TimesFn 10 (TriSectionFn (rightBankSite-Fn ?Site1)))
                         (TimesFn 250 0.75 0.8)) ;rate * factors
                         1.15 60))   ; expected time factor * 60 mins
        (pair ?I ?J)).

; gapLength could be altered, but don't account for this
F: (potentialAction flattenBank-Right
(CSVCondition Site ?Site1 gapLength (testCSV lessThan ?X))
(CSVCondition Site ?Site1 gapLength (testCSV lessThan ?X))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

; leftBankSlope is not
F: (potentialAction flattenBank-Right
(CSVCondition Site ?Site1 leftBankSlope (testCSV lessThan ?X))
(CSVCondition Site ?Site1 leftBankSlope (testCSV lessThan ?X))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

F: (potentialAction flattenBank-Right
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).
F: (potentialAction flattenBank-Right
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).
```

```
F: (potentialAction flattenBank-Right
(CSVCondition ?Class ?B objectFoundInLocation (testCSV equals ?Site1))
        (CSVCondition ?Class ?B objectFoundInLocation (testCSV equals ?Site1))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).

;--------------

; prepareFarBank

; an M88 can be used for preparation
F: (potentialAction prepareFarBank
  (conj (CSVCondition M88ArmoredRecoveryVehicle Some
objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site1)))
(CSVCondition Site ?Site1 bridgedBy (testCSV equals True)))
  (CSVCondition Site ?Site1 bridgedBy (testCSV equals True))
(consume MilitaryOpTime (MinutesDuration 30 50))
        (pair ?I ?J)).
F: (potentialAction prepareFarBank
  (conj (CSVCondition M88ArmoredRecoveryVehicle Some
objectFoundInLocation
(testCSV equals (rightRegion-Fn ?Site1)))
(CSVCondition Site ?Site1 bridgedBy (testCSV equals True)))
  (CSVCondition Site ?Site1 bridgedBy (testCSV equals True))
(consume MilitaryOpTime (MinutesDuration 30 50))
        (pair ?I ?J)).

; a Bulldozer could also be used
F: (potentialAction prepareFarBank
  (conj (CSVCondition Bulldozer Some objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site1)))
(CSVCondition Site ?Site1 bridgedBy (testCSV equals ?X)))
  (CSVCondition Site ?Site1 bridgedBy (testCSV equals ?X))
(consume MilitaryOpTime (MinutesDuration 30 50))
        (pair ?I ?J)).
F: (potentialAction prepareFarBank
  (conj (CSVCondition Bulldozer Some objectFoundInLocation
(testCSV equals (rightRegion-Fn ?Site1)))
(CSVCondition Site ?Site1 bridgedBy (testCSV equals ?X)))
  (CSVCondition Site ?Site1 bridgedBy (testCSV equals ?X))
(consume MilitaryOpTime (MinutesDuration 30 50))
        (pair ?I ?J)).

F: (potentialAction prepareFarBank
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S leftBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).
F: (potentialAction prepareFarBank
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
        (CSVCondition Site ?S rightBankSurfaceAttr (testCSV equals ?ST))
(consume MilitaryOpTime (MinutesDuration 0))
        (pair ?I ?J)).
```

```
;--------------

; prepareNearBank

Default Mt: TFPlannerMT.

; an M88 can be used for preparation - no changes are modelled
; but add leftBankSurfaceAttr = HardSurface as a condition
; and make the site explicit
F:  (potentialAction prepareNearBank
  (conj (CSVCondition M88ArmoredRecoveryVehicle Some
objectFoundInLocation
(testCSV equals (leftRegion-Fn ?S)))
  (conj (CSVCondition Site ?S leftBankSurfaceAttr
(testCSV equals HardSurface))
  ?E))
(conj (CSVCondition M88ArmoredRecoveryVehicle Some
objectFoundInLocation
(testCSV equals (leftRegion-Fn ?S)))
?E)
(consume MilitaryOpTime (MinutesDuration 30 80))
        (pair ?I ?J)).

F: (potentialAction prepareNearBank
  (conj (CSVCondition M88ArmoredRecoveryVehicle Some
objectFoundInLocation
(testCSV equals (rightRegion-Fn ?S)))
  (conj (CSVCondition Site ?S rightBankSurfaceAttr
(testCSV equals HardSurface))
  ?E))
(conj (CSVCondition M88ArmoredRecoveryVehicle Some
objectFoundInLocation
(testCSV equals (rightRegion-Fn ?S)))
?E)
(consume MilitaryOpTime (MinutesDuration 30 80))
        (pair ?I ?J)).

;; Bulldozer - can be used on any surface

F: (potentialAction prepareNearBank
  (conj (CSVCondition Bulldozer Some
objectFoundInLocation
(testCSV equals (leftRegion-Fn ?S)))
  ?E)
(conj (CSVCondition Bulldozer Some
objectFoundInLocation
(testCSV equals (leftRegion-Fn ?S)))
?E)
(consume MilitaryOpTime (MinutesDuration 30 80))
        (pair ?I ?J)).

F: (potentialAction prepareNearBank
  (conj (CSVCondition Bulldozer Some
objectFoundInLocation
(testCSV equals (rightRegion-Fn ?S)))
```

```
  ?E)
(conj (CSVCondition Bulldozer Some
objectFoundInLocation
(testCSV equals (rightRegion-Fn ?S)))
?E)
(consume MilitaryOpTime (MinutesDuration 30 80))
       (pair ?I ?J)).


;--------------

Default Mt: TFPlannerMT.

;; generalisations of useful derivations
;; sites names are replaced by variables
;; all conditions of all rules used in the
;; derivation become preconditions of the
;; derived rule

;; narrowGap plan

F: (implies (and (regionIsa ?Site0 FarInterdictionSite)
 (regionIsa ?Site NearInterdictionSite)
 (leftRegion ?Site ?LR)
 (isa ?R River)
 (between ?Site0 ?R ?LR))
(potentialAction narrowGap
; conditions
(conj (CSVCondition ArmoredVehicleLaunchedBridge Some objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site)))
(conj (CSVCondition Site ?Site leftBankSurfaceAttr (testCSV equals SoftSurface))
(conj (CSVCondition Site ?Site gapLength
(testCSV lessThan (PlusFn (Meter 17.37) (Meter 10))))
(CSVCondition Bulldozer Some objectFoundInLocation (testCSV equals ?Site0)))))
; effects
(conj (CSVCondition Bulldozer Some objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site)))
(conj (CSVCondition ArmoredVehicleLaunchedBridge Some objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site)))
(conj (CSVCondition Site ?Site gapLength (testCSV lessThan (Meter 17.37)))
(CSVCondition Site ?Site leftBankSlope (testCSV lessThan (Percent 30))))))
; resources
(conj (consume MilitaryOpTime (MinutesDuration 120 180))
(conj (consume MilitaryOpTime (TimesFn (DividesFn
(DistanceFn ?Site0 ?Site) 60) 60))
(consume MilitaryOpTime (TimesFn (DividesFn (TimesFn 10
(CrossSectionFn ?Site 15)) (TimesFn 250 0.75 0.8)) 1.15 60))))
(pair ?I ?J))).


;; useAVLB - incorporating narrowGap plan

F: (implies (and (regionIsa ?Site0 FarInterdictionSite)
 (regionIsa ?Site NearInterdictionSite)
 (leftRegion ?Site ?LR)
 (isa ?R River)
 (between ?Site0 ?R ?LR))
(potentialAction useAVLB
; conditions
(conj (CSVCondition ArmoredVehicleLaunchedBridge Some objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site)))
(conj (CSVCondition Site ?Site leftBankSurfaceAttr (testCSV equals SoftSurface))
(conj (CSVCondition Site ?Site gapLength
(testCSV lessThan (PlusFn (Meter 17.37) (Meter 10))))
(CSVCondition Bulldozer Some objectFoundInLocation (testCSV equals ?Site0)))))
; effects
(CSVCondition Site ?Site bridgedBy (testCSV equals True))
; resources
(conj (consume MilitaryOpTime (MinutesDuration 120 180))
(conj (consume MilitaryOpTime (TimesFn (DividesFn
(DistanceFn ?Site0 ?Site) 60) 60))
(conj (consume MilitaryOpTime (TimesFn (DividesFn (TimesFn 10
(CrossSectionFn ?Site 15)) (TimesFn 250 0.75 0.8)) 1.15 60))
(conj (consume MilitaryOpTime (MinutesDuration 5 10))
(consume MilitaryOpTime (MinutesDuration 30 50)))))
(pair ?I ?J))).


;; prove useAVLB - simple plan

F: (potentialAction useAVLB
;conditions
(conj (CSVCondition M88ArmoredRecoveryVehicle Some objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site)))
(conj (CSVCondition Site ?Site leftBankSurfaceAttr (testCSV equals HardSurface))
(conj (CSVCondition ArmoredVehicleLaunchedBridge Some objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site)))
(conj (CSVCondition Site ?Site gapLength (testCSV lessThan (Meter 17.37)))
      (CSVCondition Site ?Site leftBankSlope  (testCSV lessThan (Percent 30))))))
; effects
(CSVCondition Site ?Site bridgedBy (testCSV equals True))
; resources
(conj (consume MilitaryOpTime (MinutesDuration 30 80))
(conj (consume MilitaryOpTime (MinutesDuration 5 10))
(consume MilitaryOpTime (MinutesDuration 30 50))))
(pair ?I ?J)).

;; mobiliseMGB plan

F: (implies (and (regionIsa ?Site0 FarInterdictionSite)
  (regionIsa ?Site NearInterdictionSite)
  (leftRegion ?Site ?LR)
  (isa ?R River)
  (between ?Site0 ?R ?LR))
 (potentialAction mobiliseMGB
 ; conditions
 (conj (CSVCondition MGBSet Some objectFoundInLocation (testCSV equals ?Site0))
 (conj (CSVCondition Bulldozer Some objectFoundInLocation (testCSV equals ?Site0))
      (CSVCondition Site ?Site gapLength (testCSV lessThan (Meter 31.09)))))
 ; effects
(conj (CSVCondition Bulldozer Some objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site)))
(conj (CSVCondition MGBSet Some objectFoundInLocation
(testCSV equals (leftRegion-Fn ?Site)))
(CSVCondition Site ?Site gapLength (testCSV lessThan (Meter 31.09)))))
(conj (consume MilitaryOpTime (MinutesDuration 240 360))
(consume MilitaryOpTime (TimesFn (DividesFn (DistanceFn ?Site0 ?Site) 60) 60)))
(pair ?I ?J))).


;;; useMGB - incorporating mobiliseMGB subplan

F: (implies (and (regionIsa ?Site0 FarInterdictionSite)
 (regionIsa ?Site NearInterdictionSite)
 (leftRegion ?Site ?LR)
 (isa ?R River)
 (between ?Site0 ?R ?LR))
(potentialAction useMGB
; conditions
(conj
(CSVCondition MGBSet Some objectFoundInLocation (testCSV equals ?Site0))
(conj
(CSVCondition Bulldozer Some objectFoundInLocation (testCSV equals ?Site0))
(CSVCondition Site ?Site
gapLength (testCSV lessThan (Meter 31.09)))))
;effect
   (CSVCondition Site ?Site bridgedBy  (testCSV equals True))
; resources
(conj (consume MilitaryOpTime (MinutesDuration 240 360))
(conj (consume MilitaryOpTime (TimesFn (DividesFn (DistanceFn ?Site0 ?Site) 60)
60))
(conj (consume MilitaryOpTime (MinutesDuration 30 80))
(conj (consume MilitaryOpTime (TimesFn (MGBConstructionFn ?Site) 1.15))
(consume MilitaryOpTime (MinutesDuration 30 50))))))
(pair ?I ?J))).
```

J COA Grammar

The COA Grammar is a product of the Year 2 Challenge problem and, consequently, is listed below.

```
<CourseOfActionStatement> =
[<GenericMissionStatement>] <CloseBattleStatement>
[<ReserveBattleStatement>] [<SecurityBattleStatement>]
[<DeepBattleStatement>] [<RearBattleStatement>] [<FiresStatement >]
[<ObstaclesStatement>] [<RiskStatement>] [<EndStateStatement >]

<CloseBattleStatement> =
<TaskStatement0> [and <TaskStatement>]
[<DecisivePointStatement>] [<MainEffortStatement>]

<ReserveBattleStatement> =
The reserve "," <Resource> "," <TaskSpec>

<SecurityBattleStatement> =
[In the security {area, zone}] <TaskStatement>

<DeepBattleStatement> =
Deep operations will <TaskSpec>

<RearBattleStatement> =
<Resource>  <RearBattleSpec>

<RearBattleSpec> =
<PartialTaskSpec>

<RearBattleSpec> =
{<TypeOfOperationSpec> to respond, responds}
to threats <LocationSpec>
[with priority to level  <ThreatLevels> threats against
<CombatSupportAsset>]
in order to <PurposeSpec>

<ThreatLevels> =
<ThreatLevel> [<ThreatLevels1>]

<ThreatLevels1> =
{and <ThreatLevel>,  "," <ThreatLevels1>}

<ThreatLevel> =
{ I, II, III}

<FiresStatement> =
{Fires, <FireSupportAsset>} will <PartialTaskSpec>

<ObstaclesStatement> =
Obstacles will <ObstacleTask>

<ObstacleTask> =
<PartialTaskSpec> [[","] and will {<ObstacleTask >}]
```

```
<TaskStatement> =
<TaskStatement0> [and <TaskStatement>]

<TaskStatement0> =
<Resource> <TaskSpec>

<TaskSpec> =
<TaskSpec0> ["," {and, then} <TaskSpec>]

<TaskSpec0> =
[{if, when, unless} [not] <EventStatement> ","] [{on order, be prepared to}]

[<TypeOfOperationSpec> to] <Task>
[in order to <PurposeSpec>]

<PartialTaskStatement> =
<Resource> <PartialTaskSpec>

<PartialTaskSpec> =
<PartialTaskSpec0>["," { and, then } <PartialTaskSpec>]

<PartialTaskSpec> =
conducts/performs
{the Main Attack, the Main Effort, Supporting Effort <Number>}

<PartialTaskSpec0> =
[ { on order, be prepared to, is prepared to}]
<TypeOfOperationSpec> [to <Task>]
[in order to <PurposeSpec>]

<PartialTaskSpec0> =
[ { on order, be prepared to, is prepared to}]
<Task>
[in order to <PurposeSpec>]

<PartialTaskSpec0> =
[{ on order, be prepared to, is prepared to}]
<PurposeSpec>

<Task> =
<Task0> [<LocationSpec>] [<TimeSpec>]

<Task0> = "(" <Task0> ")"

<Task0> =
{ambushes <Location>,
[conducts] attack by fire,
attrits {<Unit>, <LocationOrSpec>} [{ to, by} <Number> percent] [by
destroying <Number> <UnitEchelon>],
blocks {<Unit>, <LocationOrSpec>},
turn <Unit> [<LocationSpec>],
breaches obstacles <LocationSpec>,
bypasses {obstacles <LocationSpec>, <Unit>} [to the < Direction>],
canalizes <Unit> [<Direction>],
clears {<Unit>, obstacles [<LocationSpec>], <Location >},
contains {<Unit>, <LocationOrSpec>},
counterattacks by fire,
defeats <Unit>
delays <Unit> [<Duration>],
destroys {<Unit>, <LocationOrSpec>},
disrupts {<Unit>, <LocationOrSpec>} [<Capability>],
fixes <Unit>,
follows [<Unit>] and assumes the main effort,
follows [and supports] <Unit>,
[conducts] {forward, rearward} passage of lines [through <Unit >],
interdicts {<Unit>, <LocationOrSpec>, <Capability>},
isolates {<Unit>, <LocationOrSpec>},
neutralizes {<Unit>, <LocationOrSpec>, <Capability>},
occupies <Location>,
penetrates [{<Unit>, <LocationOrSpec>}],
reinforces <Unit>,
[performs] relief in place with <Unit> <LocationSpec>,
retains <Location>,
[performs/conducts] river crossing [operations] [<Direction>]
[across <NamedLocation>],
conducts retirement,
secures <Location>,
suppress {<Unit>,<LocationOrSpec>},
screens {<Unit>, <LocationOrSpec>},
guards {<Unit>, <LocationOrSpec>},
covers {<Unit>, <LocationOrSpec>},
seizes [<Location>],
[conducts] support by fire,
[conducts] withdrawal,
[conducts] withdrawal under pressure,
<LogisticsTaskO>}


<LogisticsTaskO> =
{resupplies [<Unit>] [with <SupplyClass>],
moves <Unit> to <LocationSpec>}

<LocationOrSpec> = {<Location>,<LocationSpec>}

<GenericMissionStatement> =
<Resource> <MissionSpec>

<MissionSpec> =
[on order] [<TypeOfOperationSpec> to] <TaskSpec>
in order to <PurposeSpec>
[[","] and/then <MissionSpec>]

<TypeOfOperationSpec> =
{<OffensiveOperationSpec>,
<DefensiveOperationSpec>,
<EnablingOperationSpec>,
[conducts] operations}
[<LocationSpec>] [<TimeSpec>]
["," {and, then} <TypeOfOperationSpec>]
```

```
<OffensiveOperationSpec> =
{<TypeOfOffensiveOperation>, <OffensiveManeuverOption>}

<TypeOfOffensiveOperation> =
{<TypeOfOffensiveOperationVerb>, <TypeOfOffensiveOperationNoun >}

<TypeOfOffensiveOperationVerb> =
{attacks,
exploits,
pursues,
conducts/performs [a/an] <TypeOfOffensiveOperationNoun>}

<TypeOfOffensiveOperationNoun> =
{offensive operations,
entry operations,
movement to contact,
attack,
counterattack,
demonstration,
feint,
raid,
search & attack,
spoiling attack,
exploitation,
pursuit}

<OffensiveManeuverOption> =
{conducts a penetration [of <Unit>],
envelops enemy positions [<LocationSpec>],
turns <Unit> out of position [<LocationSpec>],
conducts a frontal attack [against enemy positions [<LocationSpec >]],
performs an infiltration [through enemy lines [<LocationSpec>]]}

<DefensiveOperationSpec> =
{<TypeOfDefensiveOperation>, <DefensiveManeuverOption>}

<TypeOfDefensiveOperation> =
{<TypeOfDefensiveOperationVerb>, <TypeOfDefensiveOperationNoun >}

<TypeOfDefensiveOperationVerb> =
{defends, conducts/performs [a/an] <TypeOfDefensiveOperationNoun >}

<TypeOfDefensiveOperationNoun> =
{defensive operations,
defense,
area defense,
mobile defense,
retrograde operations}

<DefensiveManeuverOption> =
conducts [a] {forward defense, defense in depth}

<EnablingOperationSpec> =
{<TypeOfEnablingOperationVerb>, <TypeOfEnablingOperationNoun >}
```

```
<TypeOfEnablingOperationVerb> =
{moves,
conducts/performs <TypeOfEnablingOperationNoun>}

<TypeOfEnablingOperationNoun> =
{reconnaissance operations,
counter-reconnaissance operations, security operations,
[troop] movement operations}

<PurposeSpec> =
{<EventSpec>,
{ensure, deny} [<Unit>] <Capability>,
protect {<Resource>, <Location>, <Capability>}}
[[","] and [then] <PurposeSpec>]

<EventStatement> =
<Resource> <EventSpec>

<EventSpec> =
<EventSpec0> [and <EventSpec>]

<EventSpec> =
<EventSpec1> [and <EventSpec>]

<EventSpec1> =

{enable <Task>, prevent <Task>, complete <Task>} [by <Unit>]

<EventSpec0> =
{{begins,
engages in,
sustains,
completes/accomplishes,
fails to {complete, accomplish},
interferes with}
<PartialTaskSpec> [by <Unit>],
draw <Unit> <LocationSpec> [away from {<Unit>, <LocationSpec>}],
{enables, causes} {<EventSpec>, <EventStatement>},
surprise <Unit>,
prevents <Unit> from {<EventStatement>,<EventSpec>},
gains access to {<Location>, <Resource>},
control <LocationSpec>,
observe <Unit>,
engage <Unit>,
masses combat power <LocationSpec>,
maneuver <LocationSpec>,
detects <Resource> [activity] [<LocationSpec>],
cross <LocationSpec>,
assume the main effort,
moving <LocationSpec>}

<Capability> =
[the] ability to <EventSpec>

<Resource> =
<Resource0> [{and, or} <Resource>]

<Resource0> =
{<Unit>, <CombatSupportAsset>, <OperationalAsset>} [<LocationSpec>]

<Unit> =
{ <COANamedUnit> "," [{a/an/the, our}] [{enemy, Red, Blue}] <Unit0> ["("<
COALabel>")"]
[equipped with {<VehicleType>, <WeaponType>}],
<COALabel>} [<FormationSpec>]

<Unit> =
<COANamedUnit> [<FormationSpec>]

<Unit>=
<UnitEchelon> {[<COALabel>] ,[of <COANamedUnit>]} [<FormationSpec>]


<COANamedUnit> =
{Blue, Red} <Number> [and <COANamedUnit>]

<COANamedUnit> =
unit-label-defined-in-sketch-tool [and <COANamedUnit>]

<Unit0> =
[<UnitType>] [<UnitEchelon>]
{task force,
 first echelon,
 second echelon,
main effort,
supporting effort,
defenses,
forces,
unit,
units}

<Unit0> =
<UnitType> <UnitEchelon>

<COALabel> =
[{Red, Blue}]
{Main Effort,
Supporting Effort,
Security Force,
Reserve,
TCF}
[<Number>]

<COALabel> = <COALabel> and <COALabel>

<COALabel> =
{{RED , BLUE} <UnitType> [<UnitEchelon>] [<Number>] [<LocationSpec >],
{EF/RF/BF} [<Number>]}

<UnitType> =
{infantry,
mechanized [infantry],
motorized [infantry],
air assault [infantry],
light infantry,
armor/tank,
balanced,
aviation,
armored cavalry, air cavalry, cavalry,
[combat] engineer,
assault and obstacle,
artillery}

<UnitEchelon> =
{regional,
army group,
army,
corps,
division,
brigade,
regiment,
battalion,
squadron,
company,
troop,
battery,
platoon,
detachment,
squad,
section,
team}
["(" {-,+} ")"]


<CombatSupportAsset> =
<COANamedUnit>
[<UnitEchelon>]
{<FireSupportAsset>,
<EngineerAsset>,
<AirDefenseAsset>, command and control systems}

<FireSupportAsset> =
{[close] air support,
artillery {fires, assets},
[a/the] [{long, short} duration] FASCAM [minefield],
naval gunfire support,
counterbattery operations,
<Unit>, the <SupplyClass> supply point}

<EngineerAsset> =
{ MICLIC,
AVLB,
ribbon bridge,
girder bridge,
raft system,
ACE,
SEE,
bulldozer/dozer
tactical obstacles,
protective obstacles,
[scatterable] minefields,
demolitions,
breaching asset,
 mobility asset,
<Unit>}

<AirDefenseAsset> =
{air defense,
<Unit>}

<SupplyClass> =
Class {I, II, II, IV, V, VI, VII, VIII, IX, X}

<OperationalAsset> =
 {<WeaponType>, <GenericCommodity>, <SpecializedEquipment>}

<GenericCommodity> =
 {personnel,
  weapons,
  ammunition,
  fuel,
  intelligence}

<SpecializedEquipment> =
{ stabilized gun,
    night vision gear,
    cold weather gear}

<LocationSpec> =
{{at, in, on,  adjacent, along, from, to,  toward, forward of, behind,
across, into, out of,
vicinity of/vic}
<Location>,
between <Location> and <Location>,
<Direction> of <Location>}
[and <LocationSpec>]

<Direction> =
{north, south, east, west, northeast, northwest, southeast, southwest}

<Location> =
[a/an/the] <Location0>
[and <Location>]

<Location0> =
{<GridCoordinate> ["," <GridCoordinate0>],
<ControlMeasure>,
<NamedLocation> [theater of operations],
{terrain, area} <LocationSpec>,
<RelativeLocation>, line of communication,
location of <Resource>, location of <EventSpec>, location of
<TerrainFeature>,

area bounded by <LocationList>,

<BattlefieldFrameworkLocation>}

<LocationList> = <Location> ["," <LocationList> ]

<GridCoordinate0> =
<GridCoordinate> ["," <GridCoordinate0>]

<RelativeLocation> =
{<Direction>, center}
A <NamedLocation> is a location explicitly labeled on a map or sketch.

<ControlMeasure> = [a/an/the] <ControlMeasure0> [{<Name >, <Number>}]

<ControlMeasure0> =
{<PointControlMeasure>,
<LinearControlMeasure>,
<AreaControlMeasure>}

<PointControlMeasure> =
{ambush,
checkpoint,
contact point,
coordinating point,
decision point,
linkup point,
passage point,
point of departure,
point of interest,
rally point,
release point,
start point,
<UnitEchelon> <SupplyClass> supply point}
<Number>

<LinearControlMeasure> =
{airhead line,
bridgehead line,
direction of attack,
encirclement,
forward edge of the battle area/FEBA,
limit of advance,
<Direction> boundary of the <Unit>,
boundary between <Unit> and <Unit>,
forward line of own troops/FLOT,
line of contact/LC,
final coordination line,
line of departure/LD,
line of departure is line of contact/LD/LC,
probable line of deployment/PLD,
holding line,
phase line,
release line,
coordinated fire line/CFL,
fire support coordination line/FSCL,
no fire line/NFL,
restrictive fire line/RFL,
fortified line}
[<Number>]

<AreaControlMeasure> =
{area,
area of operations [of <Unit>],
<UnitEchelon> support area, BSA, DSA, CSA,
axis of advance,
battle position,
strong point,
drop zone/DZ,
landing zone/LZ,
pickup zone/PZ,
assault position,
attack position,
attack by fire position,
support by fire position,
infiltration lane,
objective,
airhead,
bridgehead,
named area of interest,
targeted area of interest,
restrictive fire area,
area target,
fire support area,
assault crossing area,
decoy mined area [","  fenced ","],
engagement area,
assembly area,
fortified area}
[<Number>]

<BattlefieldFrameworkLocation> =
{main battle area,
close battle area,
security {area, zone},
depth,
[<Unit>] [{left,right,<Direction>}] flank,
[<Unit>] rear [area]}

<TimeSpec> =
{at <DTG>,
{not later than/NLT, not earlier than/NET} < DTG>,
prior to <DTG>,
<DurationStatement>}

<TimeSpec> =
{at <DTG>,
{not later than/NLT, not earlier than/NET} <DTG>,
prior to {<DTG>, <EventSpec>},
after {<DTG>, <EventSpec> }
<DurationStatement>}

<DurationStatement> =
{until {<DTG>, <Event>},
between {<DTG>, <Event>} and {<DTG>, <Event>},
throughout {<DTG> to <DTG>, <Event>}}

<MainEffortStatement> =
{<Unit>, <Location>} is the main effort [<TimeSpec>]

<DecisivePointStatement> =
{<EventSpec>, <Location>, <Unit>, <TimeSpec>} is the decisive point.

<EndStateStatement> =
at the conclusion of this operation "," <Resource> will <EventSpec>

<RiskStatement> =
Risk is assumed in this course of action <RiskConditions>

<RiskConditions> =
by <RiskConditions0>
[and <RiskConditions>]

<RiskConditions0> =
{ failing to protect <LocationSpec>,
not designating a TCF,
defending forward with the bulk of the <Unit> combat power,
allocating only a <UnitEchelon> of <UnitType> to <Location>,
designating only a <UnitEchelon> sized reserve,
not designating a reserve,
assigning <Unit> a disproportionately large area of operations,
assigning <Unit> responsibility for a disproportionately large number of
enemy
forces,
allowing [<Unit>] insufficient time to <Task>,
assigning many tasks to <Unit>,
conducting operations with degraded strength
[in <GenericCommodity>],
operating with insufficient tactical intelligence
[regarding {<Unit>, <PartialTaskStatement>, <Event>}],
failing to secure <Resource> against loss or damage,
assigning the main effort tasks that do not accomplish the unit overall
purpose,
assigning the key task <PartialTaskStatement> to [<Unit> ","] an aviation
unit,
assigning the reserve mission exclusively to [<Unit> ","] an aviation unit,
[<Unit>] [conducting a task/operation "("] <PartialTaskSpec> [")"] with
{insufficient combat power, an insufficient force ratio},
[<Unit>] [conducting a task/operation "("] <PartialTaskSpec> [")"] not
ideally suited for {<Unit>, this type of unit},
[<Unit>] [conducting a task/operation "("] <PartialTaskSpec> [")"] in
terrain not suited for the {operation, unit type}}
```

```
<BattlefieldEffectsStatement> =
[<EngagementAreaStatement>] [<BattlePositionsStatement>]
[<InfiltrationStatement>] [<AvenuesOfApproachStatement>]
[<KeyTerrainStatement>]

<EngagementAreaStatement> =
<Location> is a suitable engagement area.

<BattlePositionsStatement> =
<Location> is a suitable battle position for up to a <UnitEchelon>
[of <UnitType>]

<InfiltrationStatement> =
<Location> is suitable terrain for infiltration of up to a <UnitEchelon>
[of <UnitType>]

<AvenuesOfApproachStatement> =
<NamedLocation> is a <UnitEchelon>-sized avenue of approach
[and is currently {excellent, good, fair, poor, unsuitable}
for [military] operations].

<MobilityCorridorStatement> =
<NamedLocation> is a <UnitEchelon>-sized mobility corridor
[and is currently {excellent, good, fair, poor, unsuitable}
for [military] operations].

<KeyTerrainStatement> =
<NamedLocation>["," [a <TerrainFeature> at] <LocationSpec> ","]
is key terrain [because controlling it allows/affords its owner
<Capability>].

<TerrainFeature> =
{bridge,
hilltop,
intersection of <NamedLocation>,
 }

<EnemyVulnerabilitiesStatement> =
{ [<WeaponType> has a standoff range of <Number> meters against
<WeaponType>],
[{<WeaponType>, <Unit>} {lacks, possesses} <OperationalAsset> making it [
{more, less}] {effective, ineffective} [{in, during, while}
<Environment>][<LocationSpec>]] ,
[{<WeaponType>, <Unit>} has {more, less} <QualityNoun> than {<WeaponType>,
<Unit>}] ,
[{<WeaponType>, <Unit>} will be [most] vulnerable [to {<WeaponType>,
<Resource>}] {when, at}{<Location>, <FormationSpec>} {<TimeSpec>,
<EventSpec>}]}

<Environment>=
{<WeatherCondition>, <Terrain>, <EquipmentState>}

<WeatherCondition> =
{darkness,
daylight,
snow,
rain,
wind}

<Terrain> =
{desert,
mountains}

<EquipmentState> =
{moving,
stationary}

<QualityNoun>=
{speed,
accuracy,
firepower,
mobility,
stealth}[{",", and} <QualityNoun>]

<HabitualOrganizationStatement> =The habitual organization of
<AssetsAvailable>

<TaskOrganizationStatement> = The task organization of <AssetsAvailable>

<AssetsAvailable> =
{<AssetsAvailable0>, <ForcesAvailable>} [and <AssetsAvailable>]

<AssetsAvailable0> =
<Number> of <UnitType> <UnitEchelon> equipped with { <VehicleType>,
<WeaponType>}

<ForcesAvailable> =
{Blue, Red} unit ","<GenericTaskOrganization>
[{Blue, Red} unit ["("<COALabel>")" ] is subordinate to
<GenericTaskOrganization>]
[ <CombatEffectivenessStatement>]

<GenericTaskOrganization> =
<Unit> ["("<COALabel>")"] {consisting of, including} "[" <SubUnit> "]"

<SubUnit> =
{<Unit> ["("<CommandRelationshipStatement>")"]
["("<SupportRelationshipStatement>")"], <Number> of <Unit>
["("<CommandRelationshipStatement>")"]
["("<SupportRelationshipStatement>")"], <GenericTaskOrganization>}
["," <SubUnit>]

<CommandRelationshipStatement> =
<CommandRelationshipStatement0> [and <CommandRelationshipStatement>]

<CommandRelationshipStatement0> =
[<COANamedUnit> is] {OPCOM, attached, OPCON, TACON} to <COANamedUnit>

<SupportRelationshipStatement> =
<SupportRelationshipStatement0> [and <SupportRelationshipStatement> ]

<SupportRelationshipStatement0> =
[<COANamedUnit> is]
{in direct support of,
in general support of,
in general support reinforcing to,
reinforcing}
<COANamedUnit>

<CombatEffectivenessStatement> =
<Unit> is <Number> percent combat effective.

<MissionTwoUp> =
<MissionStatement>

<IntentTwoUp> =
<IntentStatement>

<MissionOneUp> =
<MissionStatement>

<IntentOneUp> =
<IntentStatement>

<MissionStatement> =
<COANamedUnit> <MissionSpec>

<AdjacentUnitStatement> =
{to our front/rear,
to our left/right,
to the <Direction>,
in the <COANamedUnit> {security area/zone, deep battle}}
"," <TaskStatement>

<SpecifiedTask> =
<PartialTaskStatement> [(Source: <SourceOfTask>)]

<ImpliedTask> =
<PartialTaskStatement> [(Inferred from: <SourceOfTask>)]

<EssentialTask> =
<PartialTaskStatement>

<ConstraintStatement> =
<NamedUnit> {must, may, must not} <PartialTaskSpec>

<KeyTimeSpec> =
<TimeSpec> "," <EventSpec>

<RecommendedFireSupportTask> =
<Resource> <FireSupportTaskSpec>

<FireSupportTaskSpec> =
[{if, when, unless} [not] <EventSpec> ","] [{on order, be prepared to}]
```

```
[<TypeOfOperationSpec> to] <FireSupportTask>
[in order to <PurposeSpec>]
[and/then <FireSupportTaskSpec>]

<FireSupportTask> =
{suppress {<Unit>, <LocationSpec>},
destroy {<Unit>, <LocationSpec>},
neutralize {<Unit>, <LocationSpec>} [<Capability>],
<InformalFireSupportTask>}

<InformalFireSupportTask> =
{block {<Unit><LocationSpec>},
canalize <Unit>,
counterattack by fire,
delay {<Unit><LocationSpec>},
disrupt {<Unit><LocationSpec>},
fix {<Unit><LocationSpec>},
attrit {<Unit><LocationSpec>}}

<EngineerCapabilityStatement> =
<EngineerAsset> can
{<MobilityTask>, <CountermobilityTask>, <TaskStatement>}

<MobilityTask> =
{breach obstacles [<LocationSpec>],
bypass obstacles [<LocationSpec>],
penetrate obstacles [<LocationSpec>],
ford river/stream [<LocationSpec>],
bridge gap/river/stream [<LocationSpec>],
perform river crossing [across <River>] [<LocationSpec>],
clear {obstacles, debris} [<LocationSpec>],
repair damage to {road, airfield} [<LocationSpec>]}

<CountermobilityTask> =
{fix a <GenericUnit> [for <Number> <TimeUnit>],
block a <GenericUnit> [for <Number> <TimeUnit>],
turn a <GenericUnit> [for <Number> <TimeUnit>]}

<ApprovedMissionStatement> =
<MissionStatement>

<IntentStatement> =
The intent is to <IntentSpec> "." [<IntentStatement>]

<IntentSpec> =
[{first, then}] <PartialTaskSpec>
[<TimeSpec>]
[as <QualityAdverb> as possible]

<IntentSpec> =
{ The following risks are acceptable <RiskStatement>,
<ConstraintStatement> }

<QualityAdverb> =
{rapidly,
soon,
decisively,
stealthily}

<DeceptionStatement> =
The deception target is {<Resource>, <Unit> commander}.
The objective of the deception plan is to <DeceptionObjectiveStatement>.
The deception story is that <DeceptionStory>.
[The following tasks will portray this story: <TaskStatement0>
[and TaskStatement]]

<DeceptionObjectiveStatement> =
{Cause the enemy to <TaskSpec>,
cause the enemy to believe that our intent is <IntentSpec>}

<DeceptionStory> =
<CourseOfActionStatement>

<SequencingStatement> =
The operation has the following phases:
Phase <Number> ":" <PhaseName> <PurposeSpec>
[<PhaseStatement>]

<PhaseStatement> =
Phase <Number> ":" <PhaseName> <PurposeSpec>
[<PhaseStatement>]


<BattlefieldOrganizationGuidance> =
[Decisive combat operations will take place <LocationSpec>]
[I see <CloseBattleStatement>]
[ { Maintain {a/an, at least {a, an}} <Unit> in reserve,
<ReserveBattleStatement> } ]
[The security {area, zone} should be <LocationSpec>]
[<SecurityBattleStatement>]
[Deep operations should <PurposeSpec>]
[<DeepBattleStatement>]
[Maintain {a/an, at least {a, an}} <Unit>
{as a TCF, for security in the <UnitEchelon> rear}]
[<RearBattleStatement>]

<CriticalEventDefinition> =
<CriticalEventName> ":" <Unit> { <Task>, <TypeOfOperationSpec>}

<CriticalEventAssertion> =
{ <CriticalEventName> before <CriticalEventName> ,
<CriticalEventName> meets <CriticalEventName> ,
<CriticalEventName> overlaps <CriticalEventName> ,
<CriticalEventName> starts <CriticalEventName> ,
<CriticalEventName> during <CriticalEventName> ,
<CriticalEventName> finishes <CriticalEventName> ,
<CriticalEventName> equals <CriticalEventName> }
<CriticalEventAssertion> = <CriticalEventAssertion> or
<CriticalEventAssertion>
```

K Implementing a Workarounds Planner in Cyc: An HTN Approach

John Kingston and Stuart Aitken

HPKB Report, May 1999

Abstract

This report describes AIAI's implementation of a workarounds planner in Cyc, as part of the DARPA-sponsored High Performance Knowledge Bases (HPKB) program. This work introduced an ontology of planning terms into Cyc that was derived from an established AI planning system, and then developed a simplified HTN planner that was applied to a real-world planning problem. The aims of this work were to develop an ontology of planning (i.e. reasoning) within an existing large ontology-based system, and to demonstrate that knowledge-based reasoning can be performed using simple reasoning techniques if the ontology is sufficiently powerful.

The work demonstrated that the ontology of planning was sufficiently powerful to support knowledge-based planning, and a simple reasoning technique (backwared chaining theorem-proving) was able to produce very accurate plans by using this ontology, although various inefficiencies and simplifications suggest that this approach would be inappropriate for the most complex planning problems.

K.1 Introduction

This paper describes AIAI's implementation of a workarounds planner in Cyc, as part of the DARPA-sponsored High Performance Knowledge Bases (HPKB) program. Cyc [8] is a system that was designed to support common-sense reasoning, and a very large ontology, describing many aspects of the world, has been developed in order to enable such common-sense reasoning. This work introduced an ontology of planning terms into Cyc that was (ultimately) derived from an O-Plan, an established AI planning system [9], and then developed a simplified HTN planner that was applied to a real-world planning problem, based on the ontology of planning. The two aims of the work were:

1. To equip Cyc with a well-justified, fully declarative representation sufficient to support knowledge based planning;
2. To experiment with developing an HTN planner in Cyc.

The potential benefits of introducing an ontology of planning into Cyc include enriching planning with common-sense reasoning about planning terms (such as reasoning about the capabilities of vehicles and the characteristics of petroleum, rather than simply characterising them as allocatable or consumable resources); and being able to guide knowledge acquisition for a planning problem by knowing what needs to be acquired. If all of these features can be realised, then the ontology constitutes a powerful resource for knowledge-based reasoning. As for the simplified HTN planner, one of the reasons for developing this was to verify that the ontology was adequate for representing and reasoning about a real-world planning task. However, the remaining reasons require careful explanation; why should we limit ourselves to Cyc's declarative reasoning abilities (specifically, those that were designed for theorem proving), and thus force ourselves to use an approach to planning that has more in common with early AI planners such as Sipe [10] than with modern AI planners, when Cyc has an underlying Lisp-like functional language (known as SubL), plus the ability to perform forward chaining, default reasoning, and hypothesizing?

The answers lie in a mixture of pragmatism and pioneering. The planning task that is tackled is the planning of workarounds i.e. how to circumvent or overcome obstacles on a road, or impassable sections of the route. This domain was chosen as one of the "challenge problems" for the HPKB program. These challenge problems, whose organisation and content are described in [3], were designed as a technology testbed, in order to demonstrate the capabilities of various AI technologies in rapid knowledge base construction, ontology development, common-sense reasoning, and problem solving. By the time AIAI were invited to participate in this challenge problem, another group of technologists was already working to develop a workarounds planner using Cyc and SubL; so it was pragmatic to choose a different approach. The pioneering element arose because, as far as we were aware, no-one had previously written an HTN planner in Cyc.

In order to achieve our goal of reasoning about planning within Cyc, we had to define an ontology of planning and an ontology of equipment. The ontology of equipment (vehicles and weapons) was supplied to HPKB particpants as part of Cyc, so that will not be discussed here. However, the most innovative feature of the workarounds planner that we developed was the ontology of planning; this was derived from the style of planning used by O-Plan [9].

The structure of this paper is as follows: The workarounds domain is treated in Section K.2 and HTN planning is introduced in Section K.3. Section K.4 describes the implementation of the planner in Cyc, focusing on the representational techniques used. Performance in the challenge problem is outlined in Section K.5, and finally, some conclusions are drawn.

K.2 The Workarounds Domain

The workaround planning challenge problem required deciding how to circumvent or overcome obstacles to traffic. Through knowledge acquisition that was performed in the course of the first year, and made available to all parties in the HPKB program, it became clear that there were six different ways of circumventing or overcoming obstacles:

- Bridging gaps;
- Filling gaps;
- Reducing obstacles until they are trafficable, or demolishing them completely;
- Finding alternate routes;
- Providing alternate transport (e.g. ferries across water);

• Clearing minefields.

Each of these classes of solution had several instances; for example, bridging a gap can be done with an AVLB (a light bridge carried on an armoured vehicle), a medium girder bridge, a Bailey bridge, or a ribbon bridge. Each solution instance has its own constraints; for example, AVLBs require both banks to be fairly level, and have a maximum usable length of about 20 metres, while ribbon bridges can only be used on water.

The planning requirements of the problem become clear when it is realised that each solution instance may require multiple steps (e.g. first transport the AVLB to the gap site, then set it up); the various constraints on solutions may require further steps (e.g. one bank must be bulldozed to make it sufficiently level before an AVLB can be set up, which requires getting a bulldozer to the site); and a full workarounds solution may make use of more than one solution instance (for example, the approach to a blown bridge may be mined, requiring both mine clearance and gap crossing; or a gap with a river in it may be crossed by bulldozing the banks on both sides to create two alternate routes and bridging the river with a ribbon bridge).

The technology developers were provided with information on the target, the damage to it, and key features of the local terrain; the units (tanks or trucks) that would be likely to use that transport route; and a detailed description of resources (such as Army engineering units) in the area that could be used to repair the damage. They were also provided with the (written and diagrammatic) results of knowledge acquisition sessions conducted over the course of the year. The expected outputs were a reconstitution schedule (an estimate of the capacity of the damaged link over time), a time line of engineering actions needed to repair the link (i.e. a plan), and a set of assets required to effect the repair. From the point of view of the technology developers, this required considering alternate plans for repairing the link, calculating which plans were the most time-effective, and presenting the full details of these plans as outputs.

AIAI's system was developed to plan workarounds that involve bridging (or, more generally, spanning) a gap. Since AIAI were not originally expected to play the role of technology developers in the HPKB program, and the decision that AIAI's experimental approach to planning in Cyc should be evaluated as part of the HPKB challenge problem was only made a few weeks before the challenge problem testing phase, no other classes of workaround were considered.

The workaround problem of spanning a gap can be solved by installing a bridging device across the gap site. A range of bridging devices are available to the engineer, and the properties of the gap site will determine which may be used. For example, a ribbon bridge can only be used if the gap site is a river, and not a dry gap. We shall use the term "spanning device" to refer to mobile and non-mobile bridging devices.

As the name suggests, the hierarchical task network approach to planning is based on the idea that actions can be specified at a number of levels of abstraction, and that composite actions can be decomposed into primitive (i.e. executable) actions. The workaround planning problem has these characteristics, because there are different instances of each solution class, which may have multiple plan steps to accomplish them. For example, the set of plan schemata includes the following:

spanGap span a gap; one of the six classes of solutions described above. This can decompose into [setupSpanningDevice], [transportSpanningDevice;setupSpanningDevice] (even at this level of abstraction, multiple plan steps may be required), or [moveAVLB;spanWithAVLB] (because an AVLB is mounted on a tank, moving it is a sufficiently different operation to transporting other types of bridge that it is best represented as a separate planning entity).

setupSpanningDevice setup a spanning device. Decomposable into four solution instances: [spanWithAVLB], [spanWithRB], [spanWithMGB], or [spanWithBB][6]

spanWithAVLB decomposable into [prepareNearBank;prepareFarBank;emplaceAVLB] (steps required to get an AVLB in place)

prepareNearBank decomposable into [bulldoze-AVLB], or [moveBulldozer;bulldoze-AVLB], or (if the bank is already a suitably shallow slope) [] (a null action).

Altogether, there are 20-30 plan schemata that may be applied to the task of spanning a gap.

It is also necessary for plan schemata to refer to properties of sites. The representation equipment, sites, and their attributes was handled by other groups within the HPKB project, using a format known as FIRE&ISE [1]. For this paper, we will present the information in a tabular format.

For the *spanning a gap* problem, the equipment we consider includes: AVLB (a tank which has been modified to hold a bridge), ribbon bridge, medium girder bridge, and Bailey bridge. These devices can be used when some or all of the following attributes hold of a particular site: gap length < maximum device capability, the river bank slope < maximum device capability, the site (a river) is wet.

Spanning Device	Max. Span (m)	Max. Bank Slope	Required Wetness
AVLB	17.37	13.5	*
ribbon bridge	6.71	*	Wet
medium girder bridge	31.09	*	*
Bailey bridge DS	21.34	*	*
Bailey bridge TS	30.48	*	*

We must also represent other information, such as setup time and load capacity. The different spanning devices have different setup times, e.g. an AVLB will take 5 minutes to setup. Setup time may be dependent on the length of the gap being spanned, e.g. a medium girder bridge of less than 10.8 m should take 30 minutes to setup, while 60 minutes will be required if the gap is longer (10.8m to 31.09m). The load bearing capacity of a bridge may also be dependent on the length of the gap.

K.3 HTN Planning

AIAI have been working in knowledge-based planning for many years, with many efforts being centred around the O-Plan project [11]. In 1995, a project was carried out to provide planning

[6]AVLB, RB, MGB and BB stand for Armoured Vehicle Light Bridge, Ribbon Bridge, Medium Girder Bridge, and Bailey Bridge respectively.

support for RAF Search & Rescue helicopters [6]; as part of that project, a CommonKADS-style inference structure was developed that represented the essential knowledge roles and inference steps involved in O-Plan style planning [7]. While O-Plan itself has a lot of functionality designed to make planning efficient, its basic planning process can be captured as follows:

- All possible plan actions are captured, and represented as *task formalisms* (TFs). Each TF is a schema representing the conditions of an action (facts that must be true in the world, or created by earlier steps in the plan, in order for that action to be feasible); the effects of an action (usually expressd as changes to facts about the world); any decompositions of that action; and other useful information, such as the time required to perform that action, and whether they must start or end before/after the start or end of another action. TFs can therefore be considered to be "declarative rules"; they represent the same knowledge that a production rule would, in the form of an object or a collection of facts.
- Plan creation is triggered by defining a goal that must be reached, or a particular action that must be achieved with certain constraints. Fulfilling this goal, or making it possible for the action to be performed, becomes the top *issue* on the plan *agenda*.
- Rules and other programming procedures are defined that match the action that needs to be performed against the set of TFs. These rules are allowed to do one of three things:
  - Perform condition matching – i.e. match the conditions of a potential action against the current world state. If there is a complete match, that action is eligible to be added to the plan; if it is added, the current world state is updated to reflect the effects of that action.
  - Perform "achieve" matching – i.e. determine that a particular action cannot currently be performed because its conditions are not satisfied, but the conditions could be satisfied if another action was added to the plan at an earlier stage. The issue of adding the "enabling" action to the plan is added to the plan agenda.
  - If the action has a list of possible sub-actions, perform decomposition on that action – i.e. remove the performing of the action from the plan agenda, and replace it with separate issues requiring the performing of each of the sub-actions.

  O-Plan continues to reason until the plan agenda is empty.

K.4 HTN Planning in Cyc

In order to perform planning in Cyc, we needed to extend the ontology with the task-related concepts which are used by HTN-style reasoners. The plan schemas and reasoning mechanisms must then be defined and implemented. These components of our solution are discussed in the following sections.

K.4.1 An Ontology for HTN Planning

The ontology that was developed is shown in Figure 7. A new term was created called *TFObject*, which was used as the "superclass" of all terms in the planning ontology. From the viewpoint

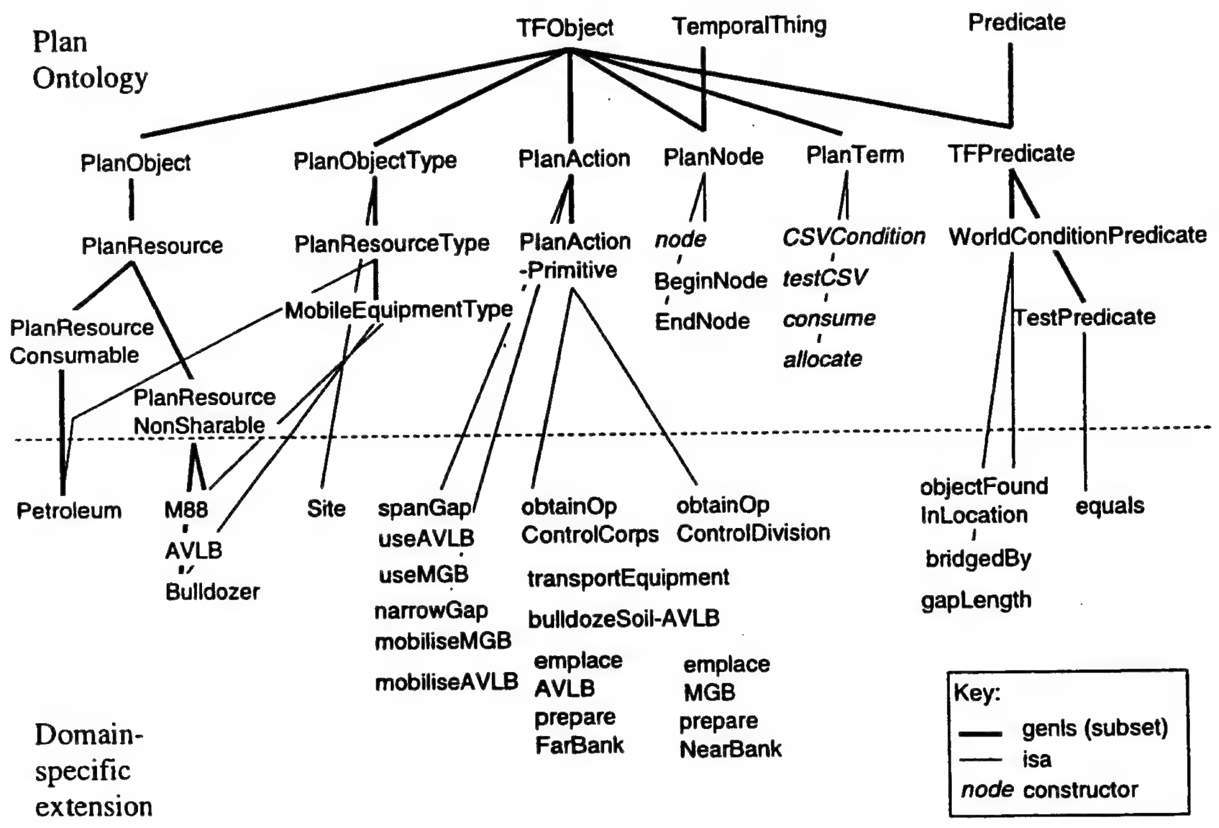


Figure 7: Ontology of HTN planning in Cyc

of the existing Cyc ontology, TFObject is just another "thing" (i.e. it is linked to the item at the top level of the Cyc upper ontology, and doesn't inherit any useful attributes). Some of the other terms do inherit useful attributes, however; for example, predicates such as *locationOf* and *bridgedBy* inherit from *Predicate*, while *PlanNode* inherits from *TemporalThing*. These inheritance links provide useful constraints to the terms in the planning ontology, as well as helping to integrate the new ontology more smoothly with the existing ontology.

The other terms created were:

PlanAction : actions that can be added to a plan. A sub-classification distinguishes decomposable actions and non-decomposable (primitive) actions.

PlanTerm : constraints that can be placed on an action; principally conditions that must hold before the action can be performed, but also amounts of resources that must be allocated or consumed. PlanTerms are also used to describe the effects of an action.

PlanResource : resources that can be created, modifed, destroyed or used during a plan. Examples include bulldozers and petroleum. In the ontology, explicit subclasses are created for consumable resources and non-sharable resources.

PlanObjectType : a second dimension for categorising plan resources – according to whether it is mobile equipment or not. The site itself (described in more detail below) is also categorised as a PlanObjectType.

PlanNode : in order to represent the plan itself as it is craeted, each action must be assigned to a node in a plan. PlanNodes inherit attributes from the pre-existing *TemporalThing* term in Cyc.

TFPredicate : Cyc requires all predicates to be explicitly declared, so predicates that were needed for the workarounds planning problem were declared under this category.

Using this ontology, a TF (which we call a *potentialAction*) can be defined as one PlanAction and a number of PlanTerms, some of which specify conditions on the action, and some of which specify the effects of the action. If the action is instantiated and used in the plan, it will hold between two PlanNodes.

It is worth describing those PlanTerms that refer to conditions in more detail. We found a need for two types of conditions: class-slot-value conditions (*CSVConditions* in the ontology) and predicate conditions (*testCSV* in the ontology). Class-slot-value conditions state that a certain slot in a certain class (or, in general, any object or instance) must have a certain value; for example, a precondition of **emplaceAVLB** is that the location (slot) of the AVLB (instance) is at the gap site (value). Predicate condition are similar, but rather than insisting on a matching value, they require a successful application of a predicate to the value; for example, *emplaceAVLB* requires that the slope (slot) of the bank (instance) be less than (predicate) 13.5 degrees.

K.4.2 Planning with a Theorem Prover

Plans are created by identifying the effect to be achieved by the plan (the final postcondition); identifying an action that will has that effect; and then asking Cyc to prove that, given the current world state, the action can be decomposed to achieve the final state. For example, if the system is required to generate a plan for spanning a gap, this goal is communicated to the planner by saying that a high-level potential action e.g. *spanGap* needs to be proved (with variables representing the gap site appropriately instantiated).

Based on the ontology of planning, HTN planning offers a choice of three options for "proving" any *potentialAction*. First, if the action's conditions are true in the current world state, the action is proved. If not, it then checks to see whether the top level action can be decomposed into sub-actions. If this is not possible, it checks to see if there are any other actions that would make the condition true. These options corresponds to condition matching, decomposition, and achieving in O-Plan. all of these options can be implemented using a theorem prover, though due to severe time constraints, only the first two were implemented in our HTN planner.

When a line of enquiry succeeds, a primitive *potentialAction* term will have its variables instantiated. Plans are represented by *potentialAction* terms that capture all the information stored in two or more smaller *potentialAction* terms. This means that eventually, an entire plan will be represented by a single term. This approach may produce large and unwieldy terms, but it also allows for a significant amount of generality in programming It also allows partial plans to be stored and used in future planning.

K.4.3 Representing Plan Schema

According to the ontology outlined above, potential actions (which correspond to O-Plan's "task formalisms") must be represented as a PlanAction, several PlanTerms, and two PlanNodes. Using the example given in Figure 8, the PlanAction is a name (*spanWithAVLB*), and the PlanTerms consists of several conditions (three in this case) and a conclusion. The PlanNodes are represented by the variables ?I and ?J in Figure 8; if **spanwithAVLB** should be added to the plan, the PlanNodes would be used to specify ordering constraints on that action.

As outlined briefly in section K.4.1, the condition terms are defined by class-slot-value terms, which specify a slot for a given plan object, and test terms, which specify a predicate test on values of slots. In Figure 8, there are three conditions: that the length of the gap is less than 17.37 metres, that the maximum slope of either riverbank is 13.5 degrees; and that the AVLB is at the gap site. The final term represents the effect of this action if it is added to the plan; the gap is considered to be bridged with an AVLB.

K.4.4 Capabilities and Limitations of the HTN Planner

The planner that we developed was capable of generating plans in the form of a single Cyc term; this was demonstrated in the challenge problem evaluation, which is described in section K.5. It had a number of limitations, however; in addition to the limitation on inference strategies, there were also simplifications (principally in the representation of time and temporal ordering)

```
F: (implies (isa ?AVLB AVLB)
    (potentialAction spanWithAVLB
       (conj
         (CSVCondition Site ?Site gapLength
           (testCSV lessThan (Meter 17.37)))
         (conj
           (CSVCondition Site ?Site riverBankMaxSlope
             (testCSV lessThan (Degree-UnitOfAngularMeasure 13.5)))
           (CSVCondition AVLB ?AVLB locationOf (testCSV equals ?Site))))
        (CSVCondition Site ?Site bridgedBy (testCSV equals ?AVLB))
        ?I
        ?J)).
```

Figure 8: Potential action for *spanWithAVLB*

due to the very short time available for development, and some other difficulties that arose as a result of the "one big term" approach to plan representation. In this section, some of the difficulties of the "one big term" approach will be discussed.

One of the difficulties with a theorem-proving approach arises because the plan is generated by a single inference query. Depending on the implementation of the reasoning system, this may make it awkward or impossible to access other useful features of the reasoning system. In the version of Cyc that we were using, we found it necessary to write rules to tell Cyc that statements of the form *(and A (and B C))* are equivalent to *(and B (and A C))*; while Cyc can easily deduce this if these are assertions in a standard Microtheory context.

K.5 Real World Test: The HPKB Challenge Problem Evaluation

When the HPKB challenge problem evaluation [3] took place, AIAI's planner (along with three other planners) was evaluated on its ability to answer a number of test questions; re-evaluated on the same questions a week later, to test the ability of the system to be updated efficiently; and was then tested on a further set of ten questions, which drew on knowledge that hadn't previously been seen, in order to test the system's ability to handle new knowledge. AIAI's system was only able to answer two of the ten questions in both tests; that it only dealt with spanning gaps, this is hardly surprising. The accuracy obtained, however, was impressive; initial results on the first test showed 80% accuracy (13% of marks were lost due to omitting two ways of spanning a gap that the system didn't yet consider, and 7% was lost due to an erroneous time calculation); by the end of the first week, the score was up to 100%. During the modification phase, it was difficult to test the system's incremental knowledge when the knowledge that was to be incremented was not implemented; despite this, similar levels of accuracy were obtained.

K.6 Conclusions

It can be seen that the "proof of concept" system has indeed proved that an HTN planner can be implemented in Cyc, using a declarative ontology of planning and Cyc's theorem-proving reasoning only. The evaluation showed that this system produced highly accurate plans and was capable of being updated fairly easily, although a number of simplifications had to be introduced to HTN planning, and there were inefficiencies and weaknesses were in the planning process. Despite these weaknesses, it appears that ontology-based reasoning is feasible for implementing knowledge-based tasks of low or medium difficulty.

References

[1] Alphatech, FIRE&ISE document `http://projects.teknowledge.com/HPKB/`

[2] Benjamins, R., de Barros, L. N., and Valente, A. Constructing planners through problem solving methods. *Proceedings of the Knowledge Acquisition Workshop 1996*, Banff 1996.

[3] Cohen P. et al, The DARPA HPKB project. IEEE Expert. To be filled in.

[4] de Barros, L. N., Valente, A., and Benjamins, R. Modelling planning tasks. *Proceedings of AIPS 1996*, pp. 11-18, 1996.

[5] de Barros, L. N., Hendler, J., and Benjamins, R. Par-KAP: A knowledge acquisition tool for building practical planning systems. *Proceedings of IJCAI 1997*, pp. 1246–1251, 1997.

[6] H. Cottam et al, ES95 Search and Rescue paper.

[7] Kingston, J., Shadbolt, N., and Tate, A. CommonKADS models for knowledge based planning. *Proceedings of AAAI 1996.* Also published as AIAI TR-199.

[8] D. Lenat, "The Dimensions of Context Space", available from the Cycorp website, http://www.cyc.com

[9] O-Plan Project Team *Task Formalism Manual v 2.3* AIAI, July 1995.

[10] SIPE reference to be added.

[11] A.Tate, O-Plan, refererence to be filled in.

[12] Schreiber A. Th., J. M. Akkermans, A.A. Anjewierden et al, Engineering and Managing Knowledge: The CommonKADS Methodology (version 1.0). Department of Social Science Informatics, University of Amsterdam. 1998.

[13] U.S. Army Engineer School *Engineer Systems Handbook* April 1997.

[14] U.S. Army *Engineer Field Data FM 5-34* July 1997.

[15] Valente, A. Knowledge-level analysis of planning systems. *SIGART Bulletin*, Vol. 6, No. 1, pp. 33–41, 1995.

L Translating COA Texts into CycL

Stuart Aitken

HPKB Report, November 1999

Abstract

This report describes the techniques used to translate structured text into logical assertions in the CycL language of the Cyc knowledge-based system. The input texts conform to a grammar which was specifically written to express US Army Courses Of Action (COAs). Another input to the interpretation task is a CycL description of a sketch of the scenario in question, interpretation therefore also involves knowledge fusion.

L.1 Introduction

This report describes the techniques used to translate structured text into logical assertions in the CycL language of the Cyc knowledge-based system. The input texts conform to a grammar which was specifically written to express US Army COAs. Courses of Action describe the plan of a military operation in terms of the tasks to be performed and purposes of executing those tasks. COAs are typically written in a stereotypical format using a limited vocabulary, and the grammar captures this structure.

COA texts are typically accompanied by a sketch of the plan of operations. The text may refer to units and locations depicted on the sketch. As it was desired to create a unified description of the COA from both of its components, it was necessary both to translate the text and integrate the interpretation with that of the sketch. A COA sketch tool (developed by John Li, at Teknowledge) was simultaneously developed and was capable of producing a CycL description from symbols and geographical features drawn on a two-dimensional map. The sketch tool is based on a commercial GIS product.

An aim of defining the COA grammar was to ease the translation from text to logic: The hypothesis was that capturing the syntactic structure of COA statements in a domain-specific grammar, as opposed to a conventional natural language grammar, would aid interpretation. We discuss this further in the Conclusions. This report continues with a description of the background to the interpretation, then the techniques used will be described and, finally, some conclusions drawn.

L.2 Background

The COA text translation work was undertaken as part of the year 2 challenge problem on the HPKB program. The aim of challenge was to demonstrate the effectiveness of knowledge-based plan critiquing. Plan critiquing was selected as an appropriate task for knowledge-based problem solving within the existing military command processes. Plan repair was not viewed as an essential element of the challenge. Critiquers based on CYC (Teknowledge/Cycorp),

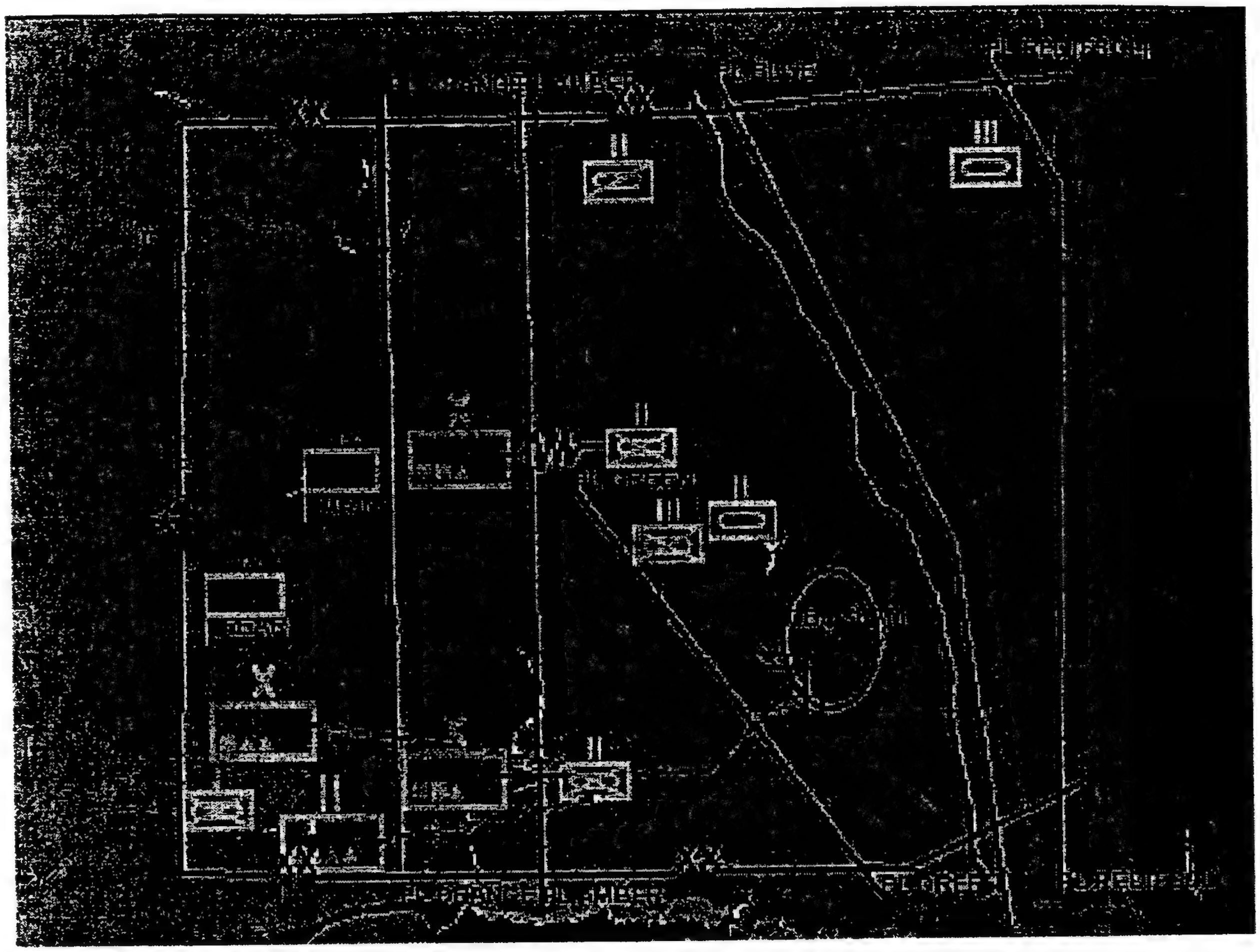


Figure 9: Sketch of a Course of Action

EXPECT (ISI), and Disciple (GMU) a simulator (UMASS), and a geographical reasoner (NWU) and were build by other partners in HPKB and were integrated on the basis of a shared ontology. These tools took the output of the sketch/text translator as their input.

L.2.1 Sketch

An example of a sketch of a Course of Action can be seen in Figure 9. Standard military symbols for units (Brigades in this case) indicate the position and type of the subunits of a Blue Division. The paths to be followed by units is also indicated. Lines known as *Phaselines* demark militarily significant boundaries and regions. Natural features such as rivers and mountains are indicated, as are battlefield obstacles such as minefields.

The sketch tool is capable of outputting a CycL representation of units, lines, features and their positions. Figure 10 shows some examples drawn form a sketch output file where 732 constants are defined and 2455 assertions are made about them. The constant *Mt_1* is defined and stated to be found in location *Loc1*. The definitions of a town and a mobility corridor are shown, and these are followed by the definition of a unit *BlueTankBn1*. The unit has eight assertions about it, specifying: speciality (armoured, mechanized infantry), allegiance (Blue), troop strength (regular), number of subunits and their type (3 armoured, 1 mechanized), and a string that is used to refer to the unit "BLUETANKBN1". The specification of the name plays an important role in interpreting the COA text, which is now described.

```
Constant: Mt_1.
F: (isa Mt_1 Mountain).

Constant: Loc1.
in Mt: BlueDivisionCOA1-1Mt.
F: (isa Loc1 GeographicalRegion).
F: (objectFoundInLocation Mt_1 Loc1).
F: (longitude (CenterFn Loc1) (Degree-UnitOfAngularMeasure 48.0536)).
F: (latitude (CenterFn Loc1) (Degree-UnitOfAngularMeasure 28.0685)).
F: (shape Loc1 (AbstractFn Square)).


Constant: NorthEasten_Town.
F: (isa NorthEasten_Town City).

Constant: MC4.
F: (isa MC4 MobilityCorridor).
F: (nameString MC4 "MC4").
F: (trafficableForUnitSize-Max MC4 Battalion-UnitDesignation).

Constant: BlueTankBn1.
F: (isa BlueTankBn1 MechanizedInfantryUnit-MilitarySpecialty).
F: (isa BlueTankBn1 ArmoredUnit-MilitarySpecialty).
F: (echelonOfUnit BlueTankBn1 Battalion-UnitDesignation).
F: (sovereignAllegianceOfOrg BlueTankBn1 Blue-Side).
F: (troopStrengthOfUnit BlueTankBn1 RegularStatus).
F: (relationInstanceExistsCount subOrgs-Direct
     BlueTankBn1 ArmoredUnit-MilitarySpecialty 3).
F: (relationInstanceExistsCount subOrgs-Direct
     BlueTankBn1 MechanizedUnit-MilitarySpecialty 1).
F: (nameString BlueTankBn1 "BLUETANKBN1").
```

Figure 10: CycL output from the Sketch Tool

L.2.2 COA Text

COA statements are written in a stereotypical form. The example below gives the text of a reserve statement which conforms to the COA grammar.

```
The reserve, BLUETANKBN1, an armor task force, follows Supporting
Effort 2, and is prepared to contain REDMECHREGT2 in order to prevent
REDMECHREGT2 from interfering with forward passage of lines through
Supporting Effort 2 by Main Effort, then follows Main Effort,
and is prepared to block REDTANKBN1 in order to enable the completion
of seizes OBJ SLAM  by Main Effort
```

The text refers to the reserve unit itself (BLUETANKBN1) and to two other blue units: Main Effort and Supporting Effort 2. These are references to the unit playing the role of the Main Effort (performing the main task of the operation) and a supporting unit. Red units are referred to by name. The area OBJ SLAM (Objective Slam) is referred to but the text gives no indication of its position. The text refers to the tasks of other units (forward passage of lines, block and seize) but does not actually specify a task for the reserve - the reserve *is prepared to* perform the tasks of contain and block. More details of the model of relations between COA elements is given later.

A small number of the grammar rules which specify legal reserve statements are given below. These show only the decomposition ReserveBattleStatement - TaskSpec - TaskSpec0, but serve to show how the COA grammar differs from a natural language approach where the categories would be NP, VP etc.

```
<ReserveBattleStatement> =
 The reserve "," <Resource> "," <TaskSpec>

<TaskSpec> =
 <TaskSpec0> ["," {and, then} <TaskSpec>]

<TaskSpec0> =
 [{if, when, unless} [not] <EventStatement> ","]
 [{on order, be prepared to}]
 [<TypeOfOperationSpec> to] <Task>
 [in order to <PurposeSpec>]
```

An example of the core text of a COA is given below. There are 9 types of statement (not all appear in this example) and in addition there are X types of statement describing other elements of the scenario such as the exact composition of the Blue and Red units, their equipment etc.

```
Mission:
BLUEMECHDIVISION1, a mechanized division attacks to seize OBJ SLAM at
130400 Aug in order to protect BOUNDARYDIVS
```

Close:
BLUETANKBGD1, an armor brigade (Main Effort) conducts forward passage of lines through Supporting Effort 2, then attacks to seize OBJ SLAM in order to prevent REDMECHDIVISION1 from gains access to the area bounded by PL BLUE, PL AMBER (LD), BOUNDARYDIVS, BOUNDARYBGDS BLUEMECHBGD1, a mechanized brigade (Supporting Effort 1) attacks in the north to fix REDMECHREGT1 and REDTANKBN1 in the area of operations of Supporting Effort 1 in order to prevent REDMECHREGT1 and REDTANKBN1 from interfering with operations by Main Effort BLUEMECHBGD2, a mechanized brigade (Supporting Effort 2) attacks in the south to penetrate REDMECHREGT2 vicinity of PL AMBER (LD) in order to enable the completion of the conduct of forward passage of lines and enable the completion of seizes OBJ SLAM by Main Effort

Reserve:
The reserve, BLUETANKBN1, an armor task force,follows Supporting Effort 2,and is prepared to contain REDMECHREGT2 in order to prevent REDMECHREGT2 from interfering with forward passage of lines through Supporting Effort 2 by Main Effort, then follows Main Effort, and is prepared to block REDTANKBN1 in order to enable the completion of seizes OBJ SLAM by Main Effort

Security:
In the security zone BLUECAVALRYBN1, an armored cavalry battalion (Security Force) screens BOUNDARYDIVN in order to enable the completion of fixes REDMECHREGT1 and REDTANKBN1 by Supporting Effort 1

Deep:
Deep operations will attrit REDARTILLERYREGT1 in order to prevent REDARTILLERYREGT1 from interfering with operations by Main Effort and prevent REDARTILLERYREGT1 from interfering with operations by Supporting Effort 2, then interdicts REDTANKBN1 in order to prevent REDTANKBN1 from interfering with operations by Main Effort, then interdicts REDTANKREGT1 in order to prevent REDTANKREGT1 from moving west of PL BLUE and interferes with operations by Main Effort and prevent REDTANKREGT1 from interfering with operations by Supporting Effort 2

Rear:
BLUEMECHTEAM1, a mechanized infantry team responds to threats in the rear area with priority to level III threats against BLUEMECHDIVISION1 the Class III supply point in order to enable resupplies with Class III

Fires:
Fires will enable penetrates REDMECHREGT2 by Supporting Effort 2

L.3 Techniques

L.3.1 COA Grammar

The original grammar describing COAs was written by Alphatec and then subsequently modified to fix errors, and extended to cover additional types of textual input such as the Products of Mission Analysis (PMA)[7]

As the grammar is extensive, detailed discussion of it will be limited to two key elements - the task and purpose components. A full listing of the COA grammer is included in this report, however, it should be noted that not all of legal grammatical expressions are translatable.

The COA grammar is specified in a formal language comprising the symbols (S)
`{ < > [ ] { } , " = }` plus terminal (T) and non-terminal (NT) words/categories [8]. The interpretation is:
`<NT> = E` denotes that `E` is a production for `NT`,
`[ E ]` denotes that `E` is optional,
`{ E1, E2 }` denotes that `E1` xor `E2` must occur,
`"S"` denotes that `S`, e.g. ",", occurs in the object language.

As an example, consider:

```
<CloseBattleStatement> =
<TaskStatement0> [and <TaskStatement>]
[<DecisivePointStatement>] [<MainEffortStatement>]
```

which has the intepretation that a `CloseBattleStatement` is composed of a `TaskStatement0`; optionally followed by 'and `TaskStatement`';
optionally followed by a `DecisivePointStatement`;
optionally followed a `MainEffortStatement`.

Task Statements

Task statements describe what actions units are to perform, for example, a unit may *seize* an area of territory, or *block* an opposing unit. Task statements assign a resource (a unit) to a task spec, and may form a conjunction using ***and***. Task specs (formally, <TaskSpec0>s) may be conditional on an event, may be ***be prepared*** tasks, may be associated with an attack or defend operation, and may have a purpose. Tasks (formally, <Task0>s) are specific types of military actions - each of the verbs ambush–withdraw has a particular semantics in the COA context and no other verbs are permissable as task descriptors. Tasks typically apply to units (red or blue), locations, or locations specs. Each Task0 production corresponds to a TacticalTask in the ontology, with the exception of "follows <Unit>" which is not considered to be a Tactical Task. Tasks which a unit stands-by to perform (*be prepared* tasks) have a considerably different translation to ordinary tasks - this is discussed in section L.3.3. Tasks may occur in purpose

[7]A full human and machine-readable version of the grammar can be found at URL: `http://www.aiai.ed.ac.uk/project/hpkb/Grammar/`

[8]The grammar of the specification language is: `WFF = T | <NT> | "S" | [ WFF ] | { WFF1 } | WFF WFF; WFF1 = WFF; WFF1 = WFF1 , WFF1;`

statements, which are now described.

```
<TaskStatement> = <TaskStatement0> [and <TaskStatement>]

<TaskStatement0> = <Resource> <TaskSpec>

<TaskSpec> = <TaskSpec0> ["," {and, then} <TaskSpec>]

<TaskSpec0> =
[{if, when, unless} [not] <EventStatement> ","] [{on order, be prepared to}]
[<TypeOfOperationSpec> to] <Task>
[in order to <PurposeSpec>]

<Task> = <Task0> [<LocationSpec>] [<TimeSpec>]

<Task0> = "(" <Task0> ")"

<Task0> =
{ambushes <Location>,
[conducts] attack by fire,
attrits {<Unit>, <LocationOrSpec>} [{ to, by} <Number> percent] [by
destroying <Number> <UnitEchelon>],
blocks {<Unit>, <LocationOrSpec>},
turn <Unit> [<LocationSpec>],
breaches obstacles <LocationSpec>,
bypasses {obstacles <LocationSpec>, <Unit>} [to the < Direction>],
...
follows [<Unit>] and assumes the main effort,
follows [and supports] <Unit>,
...
covers {<Unit>, <LocationOrSpec>},
seizes [<Location>],
[conducts] support by fire,
[conducts] withdrawal,
[conducts] withdrawal under pressure,
<LogisticsTask0>}
```

Purpose Statements

Purpose statements describe how the intended outcome of a task relates to the COA plan as a whole. Examples of purposes are: to *protect BOUNDARYDIVS* (protect a Phaseline), *enable the completion of the conduct of forward passage of lines and enable the completion of seizes OBJ SLAM by Main Effort* (enable other tasks), and *prevent REDMECHDIVISION1 from gains access to the area bounded by PL BLUE, PL AMBER (LD), BOUNDARYDIVS, BOUNDARYBGDS* (prevent access to an area). In practice, purpose statements are not as limited in expression as task statements.

The two main forms of PurposeSpec are to 'protect something' (the 3rd production in PurposeSpec) and to 'enable/prevent a task' (the first production in EventSpec0). More specific purposes are to surprise a unit, or gain access to a location (2–last productions in EventSpec0).

The name EventSpec suggests that events are to be enabled or prevented. In fact, the semantics that were developed is based on sets of states that are to be enabled/prevented. In order to make this type of reference more easy to express in the grammar, the EventSpec1 production was introduced. Semantics are discussed further in section L.3.3.

```
<PurposeSpec> =
{<EventSpec>,
 {ensure, deny} [<Unit>] <Capability>,
 protect {<Resource>, <Location>, <Capability>}}
 [[","] and [then] <PurposeSpec>]

<EventStatement> = <Resource> <EventSpec>

<EventSpec> = <EventSpec0> [and <EventSpec>]
<EventSpec> = <EventSpec1> [and <EventSpec>]

<EventSpec1> =
{enable <Task>, prevent <Task>, complete <Task>} [by <Unit>]

<EventSpec0> =
{{begins, engages in, sustains, completes/accomplishes,
  fails to {complete, accomplish}, interferes with}
  <PartialTaskSpec> [by <Unit>],
...
surprise <Unit>,
prevents <Unit> from {<EventStatement>,<EventSpec>},
gains access to {<Location>, <Resource>},
...
masses combat power <LocationSpec>,
maneuver <LocationSpec>,
detects <Resource> [activity] [<LocationSpec>],
cross <LocationSpec>,
assume the main effort,
moving <LocationSpec>}

<Capability> =
[the] ability to <EventSpec>
```

L.3.2 Parsing

A definite clause grammar (DCG) was written in Prolog for the COA grammar. A first version of the parser was generated automatically from the grammar spec., but this proved to be too inefficient in terms of the search strategy used. More importantly, as the grammar evolved, the parser needed to be updated and it is clearly inefficient to repeat hand modifications to automatically generated code. Consequently, the parser became entirely hand written as a result of grammar changes, parse tree labelling changes, or code optimisations. A total of 1606 DCG rules were written.

The to aid verification of the parser code, the grammar rules were written to directly reflect the COA grammar specification. For example, the `reserveBattleStatement` was implemented directly, see below.

```
coarseOfActionStatement(coarseOfActionStatementFn([AA,A,B,C,D,E,F,G,H,I]))
        --> optGenericMissionStatement(AA),
            closeBattleStatement(A),optReserveBattleStatement(B),
            optSecurityBattleStatement(C),optDeepBattleStatement(D),
            optRearBattleStatement(E),optFiresStatement(F),
            optObstaclesStatement(G),optRiskStatement(H),
            optEndStateStatement(I).

optReserveBattleStatement(A) -->reserveBattleStatement(A).
optReserveBattleStatement(nil) --> [].

reserveBattleStatement(reserveBattleStatementFn([A, B])) --> the_,reserve,comma,
resource__(A),comma,taskSpec(B).
```

It is also evident that optional grammar productions need to be treated efficiently, therefore, an `opt<NT>` production was also implemented. The correspondence of DCG rules with COA can also be seen in the <Task0> rules:

```
task0(task0Fn([seizes,A])) --> seizes,location(A).
task0(task0Fn([fixes,A])) --> fixes,unit(A).
task0(task0Fn([penetrates,A])) --> penetrates,optUnitOrLoc(A).
task0(task0Fn([conductsForwardPassageOfLines,A])) -->
        optConducts,forward,passage,of,lines,optThroughUnit(A).
...
seizes --> ['seizes'].
seizes --> ['seize'].
seizes --> [the,seizure,of].
```

These rules also illustrate the labelling of the parse tree through the use of the `task0Fn` and the specification of its arguments, and the introduction of non-terminals to cover the case where a COA rule has both concrete and non-terminals in the production, e.g. "seizes <Location>". Note that the concrete components of COA rules are significant, even though they do not correspond to a grammar category, and the parse tree must be designed to return these elements (i.e. the arity and concepts used in the ¡NT¿Fn must be decided upon on a case-by-case basis). As the COA grammar has no verb category, it was not possible to possible to account for tense or case in a generic way. Conventional NL resources could be made use of to solve this type of problem.

As an example of a parse tree, consider the tree derived from the following quote from the Reserve statement presented earlier:
Reserve statement: *...follows Supporting Effort 2, and is prepared to contain REDMECHREGT2 in order to prevent REDMECHREGT2 from interfering with forward passage of*

lines through Supporting Effort 2 by Main Effort... The corresponding excerpt from the parse tree is:

```
TaskSpec
  TaskSpec0
    Task
      Task0
      "follows"
        Unit
          COALabel
          "supportingEffort"
            Number
            "2"
"and"
  TaskSpec
    TaskSpec0
    "prepared"
      Task
        Task0
        "contains"
           Unit
             COANamedUnit
             "REDMECHREGT2"
      PurposeSpec
      "event"
        EventSpec
          EventSpec0
          "prevents"
            Unit
              COANamedUnit
              "REDMECHREGT2"
            EventSpec
              EventSpec0
              "interfere"
                PartialTaskSpec
                  PartialTaskSpec0
                    Task
                      Task0
                      "conductsForwardPassageOfLines"
                        Unit
                          COALabel
                          "supportingEffort"
                            Number
                            "2"
                Unit
                  COALabel
                  "mainEffort"
```

This completes the description of the form of inputs to the COA interpreter, we now discuss the models of the scenarios and their underlying ontology.

L.3.3 Ontology and Scenario Models

Modelling US Army Courses of Action began by considering a concrete scenario and attempting to describe it using the CYC upper-level ontology. Naturally, this ontology had to be extended to be able to express domain concepts more precisely. The first distinction that was made was between *tasks* and *operations*. Tasks are the limited set of tactical tasks performed by a single unit, while operations are (sets of) composite offensive or defensive actions.

The modelling decisions made can be summarised as follows (the predicates used are in parenthesis):

- tasks are the central objects in the domain, and, consequently,
- tasks have units assigned to them (unitAssignedToTask),
- tasks have purposes (taskHasPurpose),
- operations may have an associated task (taskOfOperation),
- operations have units assigned to them (unitAssignedToOperation),
- the mission has a main task which is the central action to be achieved (missionLevelTaskOfCOA),
- the mission has supporting tasks (supportingTaskOfOperation)
- actions which are not tasks are performed by a unit (performedBy),
- all tasks are implicitly subtasks of the main task (subTasks-Military),
- the phrase "task then task" implies a contiguous-after temporal order,
- tasks may act on objects (objectActedOn) or may have objects as their objective (objectiveOfTask),
- tasks may occur at a location (eventOccursAt),
- however, *be prepared* tasks are assigned to units (potentialDutyOfAgent),
- deep operation tasks and fires (which have no assigned unit) are associated with the mission (deepOperationTask, fireOperationTask).

Simple tasks, such as the fix task assigned BLUEMECHBGD1 in the COA listed above, are described by a constant which is an instance of a class: `Fix1` is the CYC constant created. The unitAssignedToTask predicate denotes the unit assignment as shown below. Purpose descriptions and *be prepared* tasks are more complex. These are modelled as relations between events or agents and situation-types (sets of situations specified by an actor and possible further specified to be a subset of a category of actions such as MilitaryInterferenceAction). The purpose of `Fix1` and the potential duty of `BlueTankBn1` are expressed formally below:

```
F: (unitAssignedToTask Fix1 BlueMechBgd1).

F: (taskHasPurpose Fix1 (prevents-SitSitType Fix1
      (CollectionSubsetFn MilitaryInterferenceAction
         (TheSetOf ?OBJ (and (or (performedBy ?OBJ RedMechRegt1)
             (performedBy ?OBJ RedTankBn1))
             (objectActedOn ?OBJ BlueTankBgd1)))))).

F: (potentialDutyOfAgent BlueTankBn1 (TheSetOf ?TASK
        (and (isa ?TASK Contain-MilitaryTask)
             (objectActedOn ?TASK RedMechRegt2))) performedBy).

F: (potentialDutyOfAgent BlueTankBn1 (TheSetOf ?TASK
        (and (isa ?TASK Block-MilitaryTask)
             (objectActedOn ?TASK RedTankBn1))) performedBy).
```

L.3.4 Interpretation

The interpretation step maps from the parse tree to the desired CycL representation. Typically, mapping rules match a pattern of the parse tree and derive an assertion, or partial interpretation, from it. Due to the size of the COA grammar parse trees, and the fact that information is distibuted over distant sub-trees, an approach was developed where sub-trees relevant to constructing specific parts of the interpretation are extracted and simplified. Matching rules are then applied to these. The simplest examples are the extraction of the unit name and echelon. The most complex are the task, purpose and state information extraction. In these cases, interpretation requires two passes, the first creates the concrete constants representing units, actions, and places, the second uses these and generates the more complex relations and set descriptions.

Additional complicating factors are: using information from the sketch in the interpretation, accounting for domain assumptions that are not expressed in the sketch or text (e.g. that all tasks are subtasks of the main task), using the definitions that are made in the text i.e. that "Supporting Effort 1" refers to BLUEMECHBGD1.

The following excerpts of Prolog code show the simplification procedure and the subsequent matching procedure. **get_task** removes the arc labels from the parse tree that do not contribute to interpretation. This turns out to be the majority of labels, e.g. the eventStatementFn and eventSpecFn labels are simply ignored, the argument structure of these functions is significant however.

```
get_task(eventStatementFn([_,I|_]),R):-
        get_task(I,R).

get_task(eventSpecFn([I,J]),[R1,R2]):-
        get_task(I,R1),
        get_task(J,R2).
```

One pattern that may result from **get_task** begins:

`[[engage, [prepared, ... ] ... ] ... ]` and a rule to interpret this pattern is given below. It can be seen that units and locations require de-referencing i.e. at this point of translation these patterns are assumed to contain the non-specific references that may occur in the text. Examples are: "the area north of...", "Supporting Effort X", or "Red forces". Finding constants (or logical specifications) for these arguments requires concrete facts about the scenario context and/or general formulations of relative locations and generic force descriptions.

In contrast, the task type `T` is an ontology term which has already been derived from the lexical input. That is, if "fix" occurs in the text, `T` will be *Fix-MilitaryTask*. The CycL predicate that relates *Fix-MilitaryTasks* to the unit they act upon is found via the `taskArgPred` Prolog predicate as this relation is dependent on the task type and the thing acted on (a unit in this case). This relation can be thought of as ontology-related parameter.

Finally, the `planToInsure-SitType` CycL predicate which is used in the target assertion takes the constant for the Blue operation as an argument, therefore this contextual information must be imported into the local interpretation context via the `get_op` Prolog predicate. The final assertion is parameterised by the terms derived as described above.

```
fuse_task([[engage, [prepared, [[T,unit,UNITREF],[location,LOCREF]|_]|_]|_],L],
        U,LastTask,R):-
       dereference_unit(UNITREF,Unit),
       dereference_location(LOCREF,Loc,_),
       taskArgPred(T,unit,Pred),
       get_op(Op),
       assert_text(['planToInsure-SitType',Op,
        ['CollectionSubsetFn',T,['TheSetOf','?OBJ',
        [and,[performedBy,'?OBJ',U],[Pred,'?OBJ',Unit],
        ['eventOccursAt','?OBJ',Loc]]]]]),
       ...
       fuse_task(L,U,LastTask,R).
```

Note that the information important to interpretation includes only two labels corresponding to COA grammar non-terminals, namely, unit and location. The original parse tree did, of course, reflect the structure of the COA grammar, but the structure of a Natural Language parse tree would have led to a similar scoping of trees/subtrees. For interpretation to succeed, the dereference functions must succeed, and while this is aided by knowing whether a unit or a location is being sought—a more sophisticated approach could operate without this information. Thus the arguments for the benefits of the COA grammar over a NL grammar are not compelling as the COA grammar does not encode semantics that are strongly related to the COA ontology.

L.3.5 System Design

As noted above, the sketch and text refer to the same objects, and names for some of these are defined in the sketch tool output. The first processing step in translation is therefore to extend the parser with the names of units and locations of the scenario in question. The second step in to create a database within the translator of the CycL description of the scenario. These two steps are illustrated as operating on the outputs of the sketch tool in Figure 11.

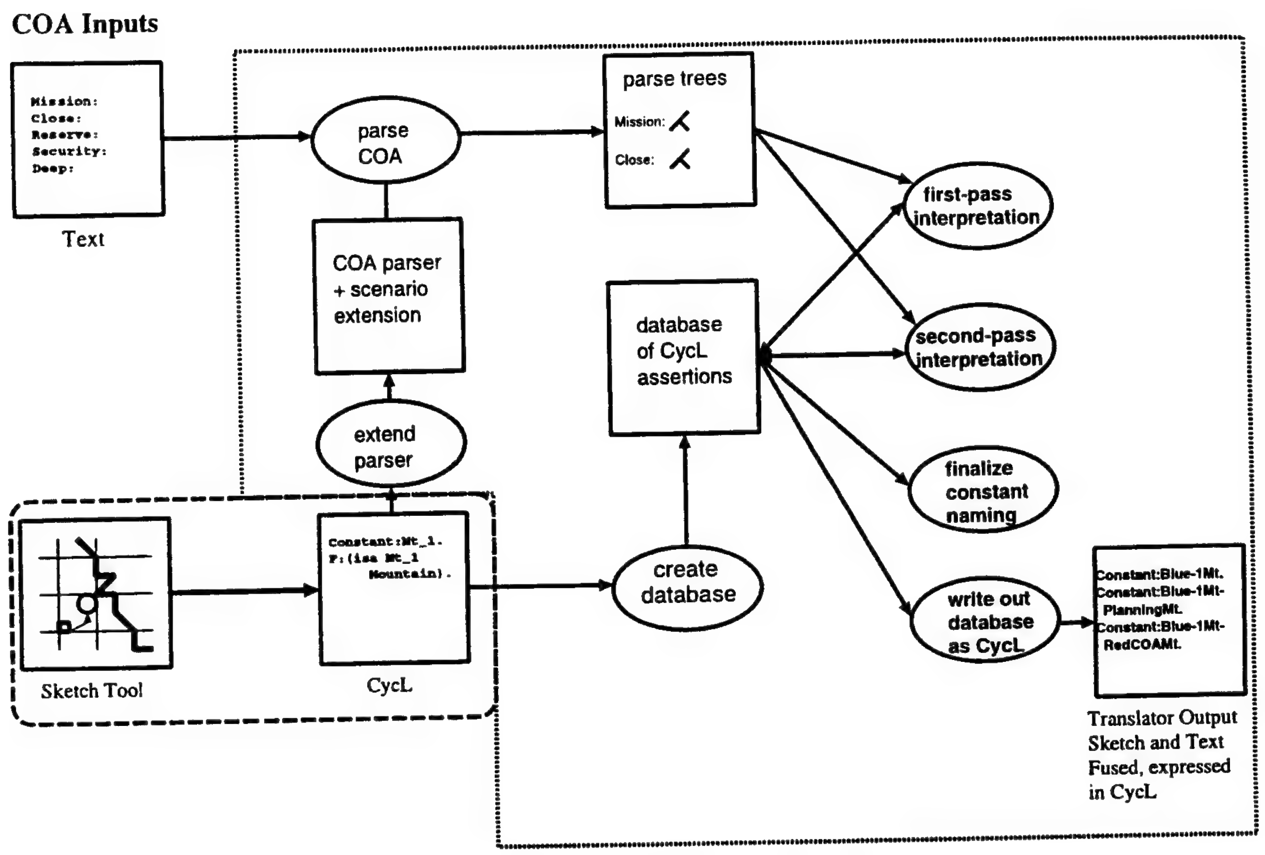

COA Inputs
Mission:
Close:
Reserve:
Security:
Deep:
Text
parse COA
parse trees
Mission:
Close:
COA parser + scenario extension
extend parser
first-pass interpretation
second-pass interpretation
database of CycL assertions
finalize constant naming
Constant:Mt_1.
F:(isa Mt_1 Mountain).
Sketch Tool
CycL
create database
write out database as CycL
Constant:Blue-1Mt.
Constant:Blue-1Mt-PlanningMt.
Constant:Blue-1Mt-RedCOAMt.
Translator Output Sketch and Text Fused, expressed in CycL

Figure 11: Translator Architecture

The COA input text is then parsed using the extended parser. This process may also cause further extensions to the parser as certain types of statement may be considered to introduce named objects. The parse trees for each statement type are then processed by the reduction and matching procedures described above. In the more complex cases, matching is perfomed again in a second pass of intepretation. The two-pass approach is required as the COA statements are processed in a sequential order and, consequently, a reference to a constant may occur prior to the processing of the statement where it is properly first introduced. For example, the purpose of task *t1* may be to enable the completion of task *t2* but the purpose phrase where this is stated may occur before *t2* has been introduced. A more general approach might use an agenda mechanism, but in practice two pases were sufficient.

The final processing steps are to rename constants that are created by default through text interpretation, such as the name of the operation, and match these to the equivalent constant - if defined - in the sketch output. These constants must be globally renamed to the common name. The contents of the database can then be written out as a text file as the database format is simply related to the format of CycL assertions and rules.

L.3.6 Results

The COA translator was used to automatically generate CycL microtheories from five different scenarios. Each scenario had major and minor variations. The inputs (measured in number of words) and outputs (measured in number of assertions) are tabulated below for the major scenario varaints. Texts of up to 2187 words were processed, resulting in the creation of 754 assertions which extended the 482 assertions created by the sketch tool.

COA	No. Words	COA Output	Sketch Output	Text Output
110	1204	993	569	424
120	1147	1000	578	422
130	1198	1014	576	438
140	1041	947	538	409
210	1879	935	328	607
220	1784		318	
230	1879		325	
240	1832	937	340	597
310	2126	1226	484	742
320	2187	1236	482	754
330	2091	1178	444	734
340	2141	1241	485	756
350	1985	1155	503	652
411	1941	948	384	564
421	1520	858	360	498
510 (RMD)	1911	929	313	616
Total	25881			8213

The sketch tool was also able to generate assertions specifying the latitude and longitude of points in the sketch. This increased the size of the sketch file from, in the case of scenario 510, 313 to 1507 assertions. This information was used by the geographical reasoner.

It is also interesting to examine the number of constants created. Again taking scenario 510 as an example, 164 constants were created in total, 90 generated from the sketch and 74 generated in addition from the text. Text interpretation therefore increased the logical model by creating eight times as many assertions as constants. This shows that the models created consisted of a relatively dense network of relations, as opposed to a set of unconnected terms.

L.4 Conclusions

An approach to structured text translation based on conventional parsing and pattern matching has been described. The problem solved by this approach is significant due to the size of the texts, the complexity of the model output, and the fact that the output was expressed in a large ontology.

We have examined the impact of the use of the COA grammar on interpretation and, contrary to intention, found that little advantage was gained that might not have been gained by other means e.g. by restricting the vocabulary.

For automated translation to be applicable in other domains it is necessary to generalise the design of the translator. We have found that the factors that parameterise the translation process include: external context, referential links within texts, structure to ontology mappings, and background assumptions about model structure. In the general case, it will also be necessary to resolve ambiguity by reference to interpretation.

Acknowledgments

This work is sponsored by the Defense Advanced Research Projects Agency (DARPA) under grant number F30602-97-1-0203. The U.S. Government is authorised to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of DARPA, Rome Laboratory or the U.S. Government.

References

[1] van Heijst, G., Schreiber, A.T., and Wielinga, B.J. Using Explicit Ontologies in KBS Development. *International Journal of Human-Computer Studies*, Vol. 46, No. 2/3, pp. 183–292.

[2] HPKB Information about the HPKB program can be found at URL: `http://www.teknowledge.com/HPKB/`

[3] Lenat, D.B. and Guha, R.V. *Building large knowledge-based systems. Representation and inference in the Cyc project.* Addison-Wesley, Reading, Massachusetts, 1990.

[4] Lenat, D.B. *Leveraging Cyc for HPKB Intermediate-level Knowledge and Efficient Reasoning*
URL: `http://www.cyc.com/hpkb/`
`proposal-summary-hpkb.html`

[5] *The Loom tutorial* Artificial Intelligence research group, Information Sciences Institute, USC, December 1995.
URL: `http://www.isi.edu/isd/LOOM/`
`documentation/tutorial2.1.html`

Knowledge representation and Reasoning in Cyc

This section describes the activities that AIAI undertook with specific relation to the Cyc ontology representation and theorem proving system.

M Extending CYC: A Summary

Stuart Aitken

HPKB Report, November 1999

This report summarises the strategies used to extend CYC in order to apply the CYC KBS and ontology to HPKB challenge problems. More detailed descriptions of the approaches can be found in the appropriate appendix.

M.1 Extending the Ontology

Our experience of extending the Upper-Level Ontology had positive and negative aspects. The tools and interfaces to the KB are very well designed, and the hyperlinked interface is likely to become a standard for this type of application. Our concerns relate to the difficulty of understanding the rational behind the design of the ontology at the lower levels. The guidance of opposing concepts such as *stuff-type* vs. *object-type* does not appear to help at this level and other guidance is lacking, see Appendix N. Issues of guidance, or, more generally, methodology, are relevant to the cooperative development of the ontology by multiple experts or knowledge engineers. For example, the experience of constructing the HPKB BattleSpace ontology showed a considerable reliance on the knowledge of the most experienced CYC user for a final judgement as to how a particular new concept should the added to the Upper-Level ontology. An important concern was to make the extension as consistent as possible with the existing ontology. That is to say, there are both issues of faithfulness to the domain being modelled, and formal ontology issues which arise when the developers have a range of expertise in CYC, and equally, a range of familiarity with the domain.

M.2 Inference in CYC: Improving Search

Our initial attempt to implement a planner in CYC relied heavily on backward chaining through a large search space (Appendix K). While the logic programming approach to representing rules that was taken may be questionable, the experience confirmed the well-known result that weak search methods don't scale up. Two solutions are possible, to implement specialised reasoners in the CYC inference machinery, or to structure search on a more conceptual basis. While specialised reasoners have been implemented for transitive reasoning, this approach is not suitable for less well specified reasoning processes such as planning. The problem-solving methods (PSM) approach offers a well-tried solution to the problem of structuring search during knowledge-based reasoning: we therefore also explored the ways in which PSMs can be implemented in CYC. Two architectures were implemented. In the first, the concepts of task-related knowledge roles, i.e. 'role in problem solving', and the domain knowledge were represented in the CYC ontology (see Appendix G). The procedural knowledge about the task structure was encoded in SubL. An alternative approach to encoding the procedural knowledge which is based on an explicit representation of this knowledge in the CYC KB has also been developed. The second approach has several advantages and will be the subject of future research.

M.3 Natural Language Input to CYC

We explored two natural language applications: the translation into CycL of COA texts expressed in a structured grammar, and the use of the Upper-Level ontology for disambiguation of general texts. The COA translator processed input texts which were in the order of 2000 words long and produced logical expressions in CycL which conformed to the BattleSpace ontology that was also developed in year 2 (see Appendix L). The translator also fused the representation of the text with that of a sketch tool which also contained infirmationabout the scenario. Our conclusions on the approach taken are that large-scale translation and fusion is possible, and need only use standard AI techniques, but is very effort-intensive. An important lesson is the need to systematise the design of such systems by considering grammar, ontology and semantic mappings as parameters. The use of a structured grammar as a specification of the input text was of doubtful value as it was very difficult to construct legal texts (although improved user input tools may help), and as the strucuture of the parse tree was only loosely related to the sematics of the phrases the complexity of the mapping task was not reduced. The use of conventional NL grammars or the improvements to structured grammars should be explored in future work.

N Extending the HPKB-Upper-Level Ontology: Experiences and Observations

Stuart Aitken

Workshop on the Applications of Ontologies and Problem-Solving Methods, (eds) Gmez-Prez, A. and Benjamins, V.R., European Conference on Artificial Intelligence 1998, Brighton, August 24-25, pp. 11-15.

Abstract

This paper describes our experience of extending the HPKB-upper-level ontology. Reuse by extension is key to reuse of generic upper-level ontologies, and we report on the use of structuring principles in this task. We argue that the documentation of design rationale is key to the reuse of this type of ontology, and that the HPKB-upper-level ontology would benefit from reorganisation.

N.1 Introduction

This paper describes an extension to the HPKB-upper-level ontology to cover *information sources* in more detail. The HPKB-upper-level ontology, which is available from the Ontolingua server [7], is intended for use in solving the "challenge problems" which have been devised as technology testbeds on the DARPA High Performance Knowledge Bases (HPKB) project [5]. AIAI are part of the HPKB programme. A major component of the first challenge problem requires searching the Web for information to answer queries about a (hypothetical) political crisis; the ability to characterise Web-based information sources in a way which identifies their ability to answer a question and the reliability of the answer is therefore important. This paper identifies concepts which would need to be represented in such an ontology, and shows how they can be implemented in Cyc.

It has been noted that there is relatively little methodological support for ontology development [1], and few reported studies on the extension of ontologies [11]. This paper describes our experience of extending an existing ontology in order to provide concrete examples of the issues and problems encountered. We then present an analysis of some of the more important issues which arise in the reuse of ontologies which define a generic upper-level conceptualisation.

Methodologies for ontology construction typically assume that a new ontology is being constructed. A middle-out approach to ontology construction has been proposed [10]. The major steps include scoping, grouping and cross-referencing concepts, producing definitions, and determining work areas. Terms in the identified work areas are then defined in middle-out fashion. It is argued that the middle-out approach avoids problems such as going into too much detail (associated with bottom-up approaches) and imposing arbitrary high-level categories (associated with top-down approaches) [10].

Project management, development-oriented, and support activities in ontology development are supported by the Methontology approach [1], which aims to specify a method for creating ontologies at a level above the language-encoding level. Ontology development includes producing a glossary of terms, and drawing diagrams such as concept classification trees and binary-relation diagrams to illustrate the connections between concepts. Terms may be drawn from other ontologies, but this is a different reuse problem to that of extending an existing ontology.

Generic ontologies which provide high-level concepts, such as event, agent, thing, and state, lack the modular structure advocated by Borst [2] and tend to have a homogeneous structure at the middle and lower levels. Terms in this type of ontology may be grouped into work areas. For example, concepts in the HPKB-upper-level ontology are grouped into 43 topical groups. These include Agents and Roles, which describe concepts and sets of relations which are central to the organisation of the ontology, as well as groups such as Emotion and Medicine which are more topic-based. In the Enterprise ontology [12], there are five work areas (all related to enterprise modelling) and, as in the HPKB-upper-level ontology, terms in each area are interrelated.

The Cyc approach to ontology development identifies a number of opposing concepts which can be used to structure the ontology. The concepts stuff-like and object-like can refer to the temporal dimension and to the nature of a substance. Events are temporally object-like, while things that exist through time, e.g. books, are temporally stuff-like as at all sub-intervals they are the same thing. However, books are object-like in nature as they cannot be subdivided and remain the same thing, unlike water, for example. We explore the use of this type of organising principle in the extension we propose in this paper. Other opposing concepts include: tangible vs. intangible, static vs. dynamic, and individual vs. collection.

Generic ontologies also differ in the degree to which they can be validated (validation is discussed further in [2]). Engineering maths and topology ontologies are capable of being validated by reference to literature in their application fields. The HPKB-upper-level, Enterprise [12] and SPAR [8, 9] ontologies do not capture knowledge in such well understood fields, therefore this form of validation is not possible. However, validation remains an important issue.

In the case study presented here, the upper ontology was already defined, and a small set of concepts were to be added. The problems we faced included the task of understanding the existing conceptualisation, but nonetheless a middle-out approach of scoping, understanding the existing ontology, then introducing intermediate level classes (i.e. classes immediately below the existing upper ontology) was productive. We describe the problems that arose in making what appeared to be 'natural' extensions to the ontology, and discuss the underlying modelling issues.

A case study of ontology extension is presented in Section N.2. This is followed by a review of the modelling decisions made in the initial extension, and those implicit in the relevant section of the upper-level ontology. Section N.4 also presents a revised ontology for information sources.

N.2 Case Study

This section presents extensions to the Cyc BaseKB which enable explicit reasoning about the sources of information that are available to the user, or to Cyc itself. The BaseKB contains the HPKB-upper-level ontology. The domain of interest was constrained to the sources of

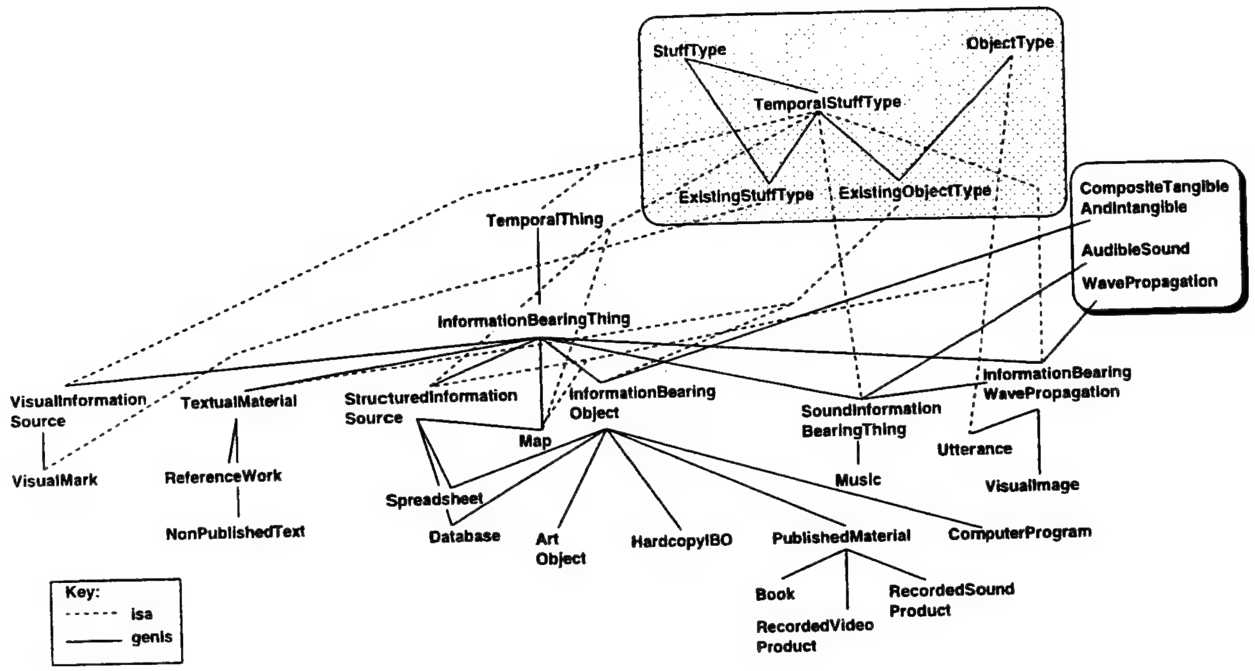


Figure 12: The existing IBT collection hierarchy

information which were identified as being relevant to solving one of the HPKB challenge problems. Our main aims were to extend the HPKB-upper-level ontology sufficiently to cover the concepts of interest, and to gain a better understanding of the modelling issues involved.

Some types of information source are already represented in the upper-level ontology, for example, books and maps. We propose a number of new sources, and a number of intermediate-level classes which characterise new source-types. A comparable ontology of documents has been posted on the Ontolingua server [7], however, many of the classes identified there already exist in the upper-level ontology (under a very different organisation).

Acquiring information may require some actions to be taken in the world. There will be some time associated with such actions, and perhaps some risks will be involved. The BaseKB contains a model of *events* which includes events that create information-bearing objects. We have reused these existing definitions to create a model of information gathering events which is integrated into the event and temporal models, but these extensions will not be presented here.

N.2.1 The Domain: Information Sources

The following information sources are representative of those used in answering challenge problem (CP) questions:

- Energy Information Administration pages
- CIA factbook

- U.S. State Department Human Rights Report
- Jane's Undersea Warfare Systems (on-line)
- the Fisher Model (a listing of air capability resources)

These information sources can be characterised by capturing the type of publication: *book*, *HTML page*, *newspaper*, *model*, and *letter*; the publication media: *hardcopy* and *softcopy*; and attributes such as *authorship*, *credibility*, *language*, and *subject area*.

It would be expected that types of publication would be modelled taxonomically, and that media would be an attribute or a property. However, this is not the case in the existing ontology and we examine these issues in detail. The attributes identified above are modelled as would be expected (by relations) and no interesting issues arise.

The information content of an information source is a distinct entity from the information source itself and is represented by *PropositionalInformationThing* (PIT) class in the upper-level ontology. The predicate *containsInformation* relates information sources to PITs. No further treatment of this issue is required for our purposes.

N.3 Relevant Upper-Level Collections

The most relevant collection containing information sources is *InformationBearingThing* (IBT). The most relevant collections containing events are *Actions* and *InformationTransferEvents*. Some useful predicates which connect these are: *products* which can take a InformationTransferEvent and an IBT as arguments, to state that the IBT is the product of the event, and *duration* which holds of an event and the time the event lasted for. Assuming that we can represent typical examples of information gathering events, and their typical durations, these classes and predicates provide a means of representing noth objects and processes in information gathering.

N.3.1 Information Bearing Things

InformationBearingThings are categorised according to whether they are textual, structured, visual, or whether they are objects. An IBT may belong to several of these classes. Figure 12 shows the *genls* (subset) relations for the IBT collection. This diagram also shows the *genls* and *isa* links between IBT collections and other upper-level collections: these will be discussed in more detail later.

A number of collections of IBTs have an obvious meaning and could simply be instantiated to represent information sources in any of the HPKB challenge problem domains, for example: *Book*, *ComputerProgram*, *Database* (e.g. the Cyc BaseKB), *Map*, *RecordedSoundProduct*, *RecordedVideoProduct*, *Spreadsheet*, *Utterance*, *VisualImage*.

This list of information sources does not include all the concepts we require, for example, HTML pages are not included. In addition, this list does not make all the distinctions we might require, e.g. Books may be in paper-copy only, or also available in some electronic form.

N.3.2 New InformationBearingThings

The existing upper-level concepts of Book and Database are organised initially by what appears to be a concept of organisational form, i.e. textual, structured, visual, and the object/stuff-like distinction. For example, *InformationBearingObjects* is an instance of *ExistingObjectType* which means that it is a collection of spatially object-like things (i.e. things which are indivisible), but is temporally stuff-like, meaning that its instances exist through time.

The definition of ExistingObjectType begins: *"A collection of collections. Each element of each element of ExistingObjectType is temporally stufflike yet is objectlike in other ways, e.g., spatially. Any one of many timeSlices of a copy of 'Moby Dick' sitting on your shelf is still a copy of 'Moby Dick' sitting on your shelf. Most tangible objects are temporally stufflike in this fashion. That book is, of course, not spatially stufflike; spatially, it is objectlike: if we take a scalpel and slice the book into ten pieces, each piece is not a copy of 'Moby Dick'. [...]"* (Copyright 1995, 1996, 1997 Cycorp. All rights reserved.)

Not all IBTs are spatially object-like as *VisualMarks* are spatially stuff-like. All IBTS are temporally stuff-like and none are temporally object-like.

At the second level of decomposition, concepts such as 'being published' and 'in hardcopy form' are introduced as collections. Distinct types of publication are then introduced. Concepts are used to model types of information source and their properties. Noting this, we will adhere to this approach as far as possible, and will postpone criticisms of the hierarchy structure until Section N.4.

Two new subcollections of Information Bearing Object (IBO) are introduced into the existing hierarchy in order to represent the new domain concepts: *SoftcopyIBO* and *Message*. Figure 13 shows the *genls* relations of the new collection hierarchy. SoftcopyIBO is introduced as a counterpart to HardcopyIBO. The natural place to locate this class is below IBO. Specifying this collection enables a distinction to be made between the electronic and paper versions of information bearing objects. Without such a collection it would not be possible to state the publication medium of IBOs such as HTML pages. The Message collection contains IBOs such as letters that are written for an identified reader. The recipients may constitute a group, in which case the IBO would be considered to be published. This collection allows a distinction to be made between letters and email, and other unpublished textual or electronic material. Messages have the spatially object-like property of IBOs in a similar way to Books. However, as they may be unpublished this collection cannot be located under Published Material and has been located directly below IBO.

The new subcollections of IBO allow HTMLPages to be defined as published material in soft-copy. *PlainHTMLPages* are essentially textual, and hence this is a specialisation of HTMLPage. *Letters* and *Email* are IBOs which are textual, and have an identified recipient. They differ as to their publication medium. Email that is circulated, and letters that are published become published textual material, as opposed to unpublished text.

The concepts of published material and non-published text must be mutually exclusive. As a result we are led to define different classes for *Letters* and *PublishedLetters* where otherwise we might say that only an attribute of the object (the letter) changes on publication. This

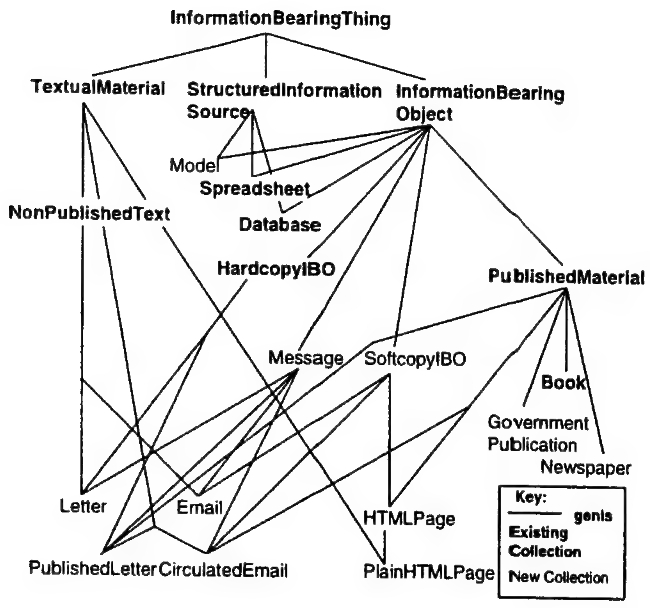


Figure 13: The extended IBT collection hierarchy

situation arises from the extensions which were introduced on largely intuitive grounds, and from the existing hierarchy structure where published material and non-published text are concepts which hold of types of publication. One remedy is to retract the `(genls Letter NonPublishedText)` assertion and allow instances of letter to be non-published text or published material as appropriate. However, the more general issues of how to structure the ontology to make it more understandable, and more amenable to extension need to be addressed.

N.4 A critique of the IBT ontology

The foregoing discussion suggests that it might be beneficial to review the structure of the existing ontology of information sources, prior to extending it. The HPKB-upper-level ontology is not composed from modules and it does not appear feasible to introduce this type of organisation. However, the structuring principles and the rationale for the design of the IBT ontology can be examined and clarified, and some validation given for the concepts used.

The principles underlying the structure of the existing hierarchy are not clear. At the first level of decomposition of IBT there are seven classes. Three of these (InformationBearingObject, SoundInformationBearingThing and InformationBearingWavePropagation) generalise to classes outside of IBT, see the shadowed box in Figure 12, and so have distinguishing features. Of these three classes, one is a subclass of another - indicating some redundancy in the *genls* definitions. Of the four remaining classes, Map and StructuredInformationSource have the same *isa* links and Map is a subset of StructuredInformationSource.

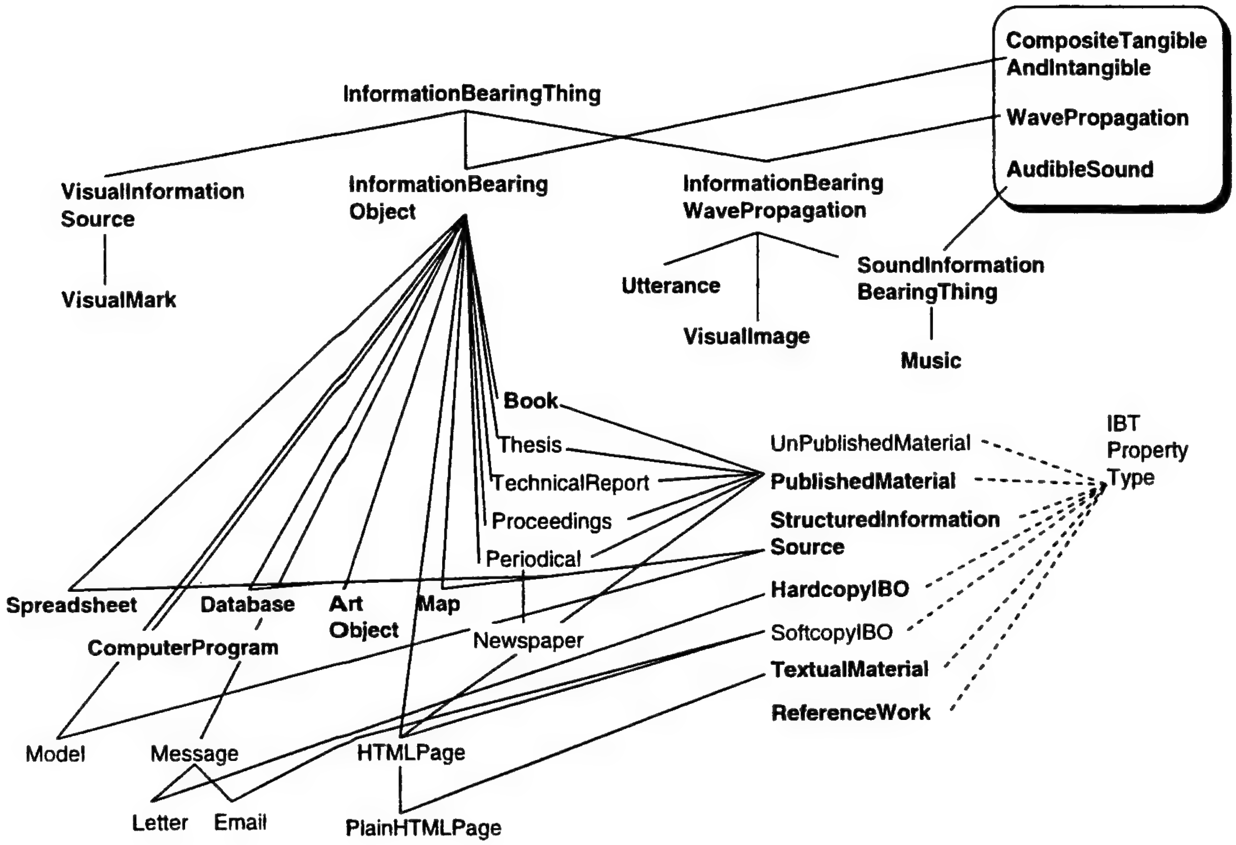


Figure 14: A proposal for restructuring the IBT hierarchy

TextualMaterial and InformationBearingObject (IBO) are of type ExistingObjectType. This in turn generalises to both TemporalStuffType and ObjectType. Map and StructuredInformationSource are stated to be of types TemporalStuffType and ObjectType - types which they would appear to belong to if they had also been defined to be of ExistingObjectType. In summary, the classes Map, StructuredInformationSource, and TextualMaterial appear to make no distinctions which are not made in InformationBearingObject. With the exception of Map, the subclasses of StructuredInformationSource generalise to IBO, indicating that there should probably be a subset relation between them. One subclass of TextualMaterial generalises to IBO, while the others do not. It is hard to explain why these subclasses do not have the IBO properties.

The VisualInformationSource class is distinguished by its lack of properties: it is not an ExistingObjectType and does not generalise to any class outside IBT. This class is only stated to be of TemporalStuffType. The definitions allow the subclass VisualMark to be existing-stuff-like in nature (note that ExistingStuffType and ExistingObjectType form a partition). In general, IBTs are not spatially stuff-like as (presumably) dividing an IBT into pieces will also divide the information encoded there.

In conclusion, there appear to be three distinct subclasses of IBT: InformationBearingWavePropagation, InformationBearingObject, and VisualInformationSource.

We suggest a reorganisation of the IBT heirarchy. The three distinct subclasses of IBT identified above form the first level of decomposition. Publication kinds are subclasses of InformationBearingObject. The kinds of IBOs can be reorganised as a taxonomy, and properties can be defined as concepts. The proposed revision of the IBT hierarchy is shown in Figure 14. Concepts may be an inherent part of the definition of a publication type, e.g. books must be published, and email must be softcopy. Exclusive properties such as being published, or not, should not be assigned to certain publication types if that property may change, e.g. subclasses of Message cannot be unpublished by definition (as noted above).

Figure 14 includes the publication types of the original IBT ontology, plus those introduced above. Additional classes are taken from the Documents ontology [7], namely: Thesis, TechnicalReport, Proceedings, and Periodical (subclasses of these are not illustrated for reasons of clarity). The opportunity of reorganising the IBT hierarchy allows the structure of that ontology to be adopted in part.

The collection *IBTPropertyType* is introduced as a collection of collections of IBTs. The members of this class are collections which define properties such as publication media, published or unpublished, textual and structured. The names of these properties in original ontology have retained where possible.

The design rationale of the proposed hierarchy has been stated, and examples given. Further, validation of the ontology is made possible by stating the origin of the terms used. By these means we believe we have increased the possibility that others could make additional extensions to the IBT ontology, and do so in a systematic way.

N.5 Discussion

The experience of reusing the HPKB-upper-level ontology has highlighted a number of issues. The opposing concepts stuff-like vs. object-like are applicable to the spatial and temporal descriptions of information bearing things. However, all IBTs are temporally stuff-like and the majority of classes are spatially object-like. The stuff-object dimension is not useful as an organising principle at the level of the ontology we considered. This is not to dispute its use at higher levels, or in other regions of the ontology.

It was apparent that concepts were used to model both properties and kinds of information sources within the same hierarchy. An attempt to extend this ontology led to modelling errors. Further examination of the ontology revealed some redundancies in the *genls* and *isa* structure. These are not significant in operational terms, but further complicate the task of understanding the structure and design rationale of the ontology.

We have considered generic ontologies which formalise consensus, or common-sense knowledge, i.e. knowledge which is not drawn from established fields of study, and which cannot be formally verified. We conclude that statements of principles, or guidelines, of ontology design, along with the provision of information which allows some validation of the ontology are particularly important for the reuse and extension of this type of generic ontology.

Acknowledgements

This work is sponsored by the Defense Advanced Research Projects Agency (DARPA) under grant number F30602-97-1-0203. The U.S. Government is authorised to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of DARPA, Rome Laboratory or the U.S. Government.

References

[1] Blázquez, M., Fernández, M., Garcia-Pinar, J.M., and Gómez-Pérez, A. Building Ontologies at the Knowledge Level using the Ontology Design Environment. *Proceedings of KAW'98 Eleventh Workshop on Knowledge Acquisition, Modeling and Management*, Banff, 1998. URL: `http://ksi.cpsc.ucalgary.ca/KAW/KAW98/blazquez/`

[2] Borst, P., Akkermans, H., Top, J. Engineering Ontologies. *International Journal of Human-Computer Studies*, Vol. 46, No. 2/3, pp. 365–406.

[3] Gruber, T.R. and Olsen, G.R. An Ontology for Engineering Mathematics. *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Doyle, J., Torasso, P., and Sandewall, E. (Eds.), Morgan Kaufmann, 1994. URL: `http://www-ksl.stanford.edu/knowledge-sharing/ontologies/html/engineering-math.text.html`

[4] van Heijst, G., Schreiber, A.T., and Wielinga, B.J. Using Explicit Ontologies in KBS Development. *International Journal of Human-Computer Studies*, Vol. 46, No. 2/3, pp. 183–292.

[5] HPKB Information about the HPKB program can be found at URL: `http://www.teknowledge.com/HPKB/`

[6] Lenat, D. *Leveraging CYC for HPKB Intermediate-level Knowledge and Efficient Reasoning* URL: `http://www.cyc.com/hpkb/proposal-summary-hpkb.html`

[7] Ontolingua Ontology Server has URL: `http://www-ksl-svc.stanford.edu`

[8] Shared Planning and Activity Representation (SPAR) URL: `http://www.aiai.ed.ac.uk/~arpi/spar`

[9] Tate, A. Roots of SPAR. to appear in the *Knowledge Engineering Review, Special Issue on Putting Ontologies to Use*, (eds.) Tate, A., and Uschold, M.F., 1998.

[10] Uschold, M. and Gruninger, M. Ontologies: principles, methods and applications. *Knowledge Engineering Review*, Vol. 11:2, 1996, pp. 93–136.

[11] Uschold, M., Clark, P., Healy, M., Williamson, K., and Woods., S. An Experiment in Ontology Reuse. *Proceedings of KAW'98 Eleventh Workshop on Knowledge Acquisition, Modeling and Management*, Banff, 1998.
URL: `http://ksi.cpsc.ucalgary.ca/KAW/KAW98/uschold/`

[12] Uschold, M., King, M., Moralee, S., and Zorgios, Y. The Enterprise Ontology. to appear in the *Knowledge Engineering Review, Special Issue on Putting Ontologies to Use*, (eds.) Tate, A., and Uschold, M.F., 1998. Also available from AIAI as AIAI-TR-195,
URL:`http://www.aiai.ed.ac.uk/~entprise/`

O Word Sense Disambiguation Using Common Sense Knowledge

Bruce Eddy

Summary of a thesis accepted for Master of Science degree, School of Artificial Intelligence, Division of Informatics, University of Edinburgh, September 1999

Abstract

Two methods for applying the Cyc knowledge base to the problem of word sense disambiguation are described. The first uses Cyc to rule out certain possible senses for a polysemous word on the grounds that their use would violate constraints imposed by the surrounding text and Cyc's "Common Sense" definitions. The second uses Cyc's knowledge base as a semantic network. It is assumed that the intended sense of a polysemous word has a closer semantic relationship to other concepts in the context than do the unintended senses. It is concluded that the first method requires a substantial investment of effort to be made general, and that the structure of the knowledge base limits the success of the second.

O.1 Introduction

Word sense disambiguation is the process of determining which sense of a polysemous word is intended in a given context. Cyc[9] is a very large knowledge base which aims to represent and to make inferences about "Common Sense" knowledge. Its size offers the possibility of overcoming the difficulties involved in scaling knowledge driven disambiguation systems to go beyond a restricted domain. This possibility is investigated in two ways.

O.2 Constraints on Word Selection

A procedure for converting natural language sentences into formulae which express their meanings in Cyc's representation language was implemented. This program was an extension of an already existing Prolog implementation of a form of Montague semantics. Using it, a sentence containing a single polysemous noun was translated into formulae in Cyc's representation language, one formula for each of selected senses of the noun. Cyc was then queried as to which of the formulae did not imply contradictions of formulae in its knowledge base. The sense corresponding to this formula was selected as the intended sense. This method correctly determined which of two senses of *plane* is intended in *The man flew the plane.*

[9]D B Lenat and R V Guha *Building Large Knowledge Based Systems* Reading, MA: Addison-Wesley (1990)

O.3 Semantic Network

The Cyc knowledge base is regarded as a semantic network[10] by mapping it onto a multigraph. Under the mapping, the representations in Cyc of entities are mapped to nodes in the graph and true formulae involving these representations are mapped to links between the nodes.

The following hypothesis is formulated: the distances (in terms of number of links) from each of the nodes which represent the unintended senses of a polysemous noun to the nodes which represent its context (suitably defined) words are less than the distances from the node which represents the intended sense to these same context nodes. To test it, a program was written which performs a breadth first search which radiates from each of the nodes which represents a context word or a sense of the polysemous word. The aim of the search is to find paths between these nodes. The program traced its progress so that if any such paths were found they could be examined and their distances recorded. It is found that the structure of Cyc makes its use as a semantic network problematic.

O.4 Conclusions

It is concluded that the first method would require a very large investment of effort to be made general. In particular, translations from each word sense in a natural language to a suitable abstraction of its representation by Cyc would have to be created. However, the existence of Cyc greatly reduces the amount of work required to create a system of this type. It is suggested that this method would be the more effective of the two were this done.

It is also found that the difficulties in separating meta- from world-knowledge limits the success of the second approach. The program often calculates distances via links which represent the structure of Cyc rather than knowledge about the world. Such links may join semantically unrelated concepts and so introduce significant distortions into the distance measurements.

O.5 Summaries of Preceding Sections

O.1 It is proposed that the comprehensiveness of Cyc's knowledge base can be utilised to expand the domain for which certain NLP problems may be solved.

O.2 A method which rules out certain possible senses for a polysemous noun on the grounds that their use would violate constraints imposed by the surrounding text and Cyc's "Common Sense" definitions is described.

O.3 A method which calculates the number of Cyc inferences needed to get from the various senses of a polysemous word to its context words is described.

O.4 It is concluded that the method of section O.2 is the more hopeful of the two but requires more work; and that the method of section O.3 is limited by the structure of the Cyc knowledge base.

[10]M R Quillian *Semantic Memory* in M L Minsky, ed. *Semantic Information Processing* pp227—259 MIT Press (1968)

DISTRIBUTION LIST

addresses	number of copies
WILLIAM E. RZEPKA AFRL/IFTD 525 BROOKS ROAD ROME, NY 13441-4505	5
DR. JOHN KINGSTON UNIVERSITY OF EDINBURGH 80 SOUTH BRIDGE EDINBURGH EH1 1HN UK	5
AFRL/IFOIL TECHNICAL LIBRARY 26 ELECTRONIC PKY ROME NY 13441-4514	1
ATTENTION: DTIC-OCC DEFENSE TECHNICAL INFO CENTER 8725 JOHN J. KINGMAN ROAD, STE 0944 FT. BELVOIR, VA 22060-6218	1
DEFENSE ADVANCED RESEARCH PROJECTS AGENCY 3701 NORTH FAIRFAX DRIVE ARLINGTON VA 22203-1714	1
ATTN: NAN PFRIMMER IIT RESEARCH INSTITUTE 201 MILL ST. ROME, NY 13440	1
AFIT ACADEMIC LIBRARY AFIT/LDR, 2950 P.STREET AREA B, BLDG 642 WRIGHT-PATTERSON AFB OH 45433-7765	1
AFRL/HESC-TDC 2698 G STREET, BLDG 190 WRIGHT-PATTERSON AFB OH 45433-7604	1

ATTN: SMDC IM PL 1
US ARMY SPACE & MISSILE DEF CMD
P.O. BOX 1500
HUNTSVILLE AL 35807-3801

COMMANDER, CODE 4TL000D 1
TECHNICAL LIBRARY, NAWC-WD
1 ADMINISTRATION CIRCLE
CHINA LAKE CA 93555-6100

CDR, US ARMY AVIATION & MISSILE CMD 2
REDSTONE SCIENTIFIC INFORMATION CTR
ATTN: AMSAM-RD-OB-R, (DOCUMENTS)
REDSTONE ARSENAL AL 35898-5000

REPORT LIBRARY 1
MS P364
LOS ALAMOS NATIONAL LABORATORY
LOS ALAMOS NM 87545

ATTN: D'BORAH HART 1
AVIATION BRANCH SVC 122.10
FOB10A, RM 931
800 INDEPENDENCE AVE, SW
WASHINGTON DC 20591

AFIWC/MSY 1
102 HALL BLVD, STE 315
SAN ANTONIO TX 78243-7016

ATTN: KAROLA M. YOURISON 1
SOFTWARE ENGINEERING INSTITUTE
4500 FIFTH AVENUE
PITTSBURGH PA 15213

USAF/AIR FORCE RESEARCH LABORATORY 1
AFRL/VSOSA(LIBRARY-BLDG 1103)
5 WRIGHT DRIVE
HANSCOM AFB MA 01731-3004

ATTN: EILEEN LADUKE/D460 1
MITRE CORPORATION
202 BURLINGTON RD
BEDFORD MA 01730

OUSD(P)/DTSA/DUTD ATTN: PATRICK G. SULLIVAN, JR. 400 ARMY NAVY DRIVE SUITE 300 ARLINGTON VA 22202	1
AFRL/IFT 525 BROOKS ROAD ROME, NY 13441-4505	1
AFRL/IFTM 525 BROOKS ROAD ROME, NY 13441-4505	1

MISSION
OF
AFRL/INFORMATION DIRECTORATE (IF)

The advancement and application of information systems science and technology for aerospace command and control and its transition to air, space, and ground systems to meet customer needs in the areas of Global Awareness, Dynamic Planning and Execution, and Global Information Exchange is the focus of this AFRL organization. The directorate's areas of investigation include a broad spectrum of information and fusion, communication, collaborative environment and modeling and simulation, defensive information warfare, and intelligent information systems technologies.